AWS Well-Architected Framework

Machine Learning Lens

Machine Learning Lens: AWS Well-Architected Framework

Table of Contents

Machine Learning Lens

Publication date: **July 5th, 2023** (*Document history*)

In recent years, machine learning (ML) has moved from research and development to the mainstream, driven by the increasing number of data sources and scalable cloud-based compute resources. AWS' customers currently use AI/ML for a wide variety of applications such as call center operations, personalized recommendations, identifying fraudulent activities, social media content moderation, audio and video content analysis, product design services, and identity verification. Industries using AI/ML include healthcare and life sciences, industrial and manufacturing, financial services, media and entertainment, and telecom.

Machine learning, through its use of algorithms to find patterns in data, can bring considerable power to its customers and thus recommends responsibility in its use. AWS is committed to developing fair and accurate AI and ML services and providing you with the tools and guidance needed to build AI and ML applications responsibly. For more information on this important topic, refer to AWS' Responsible AI.

This whitepaper provides you with a set of proven best practices. You can apply this guidance and architectural principles when designing your ML workloads, and after your workloads have entered production as part of continuous improvement. Although the guidance is cloud- and technology-agnostic, the paper also includes guidance and resources to help you implement these best practices on AWS.

Introduction

The AWS Well-Architected Framework helps you understand the benefits and risks of decisions you make while building workloads on AWS. By using the Framework, you learn operational and architectural best practices for designing and operating workloads in the cloud. It provides a way to consistently measure your operations and architectures against best practices and identify areas for improvement.

Your ML models depend on the quality of input data to generate accurate results. As data changes with time, monitoring is required to continually detect, correct, and mitigate issues with accuracy and performance. This monitoring step might require you to retrain your model over time using the latest refined data.

While application workloads rely on step-by-step instructions to solve a problem, ML workloads enable algorithms to learn from data through an iterative and continuous cycle. The ML Lens complements and builds upon the Well-Architected Framework to address this difference between these two types of workloads.

This paper is intended for those in a technology role, such as chief technology officers (CTOs), architects, developers, data scientists, and ML engineers. After reading this paper, you will understand the best practices and strategies to use when you design and operate ML workloads on AWS.

Run a Machine Learning Lens review in your AWS account

A common request from our customers has been to enable them to run a *self service* Machine Learning (ML) Lens review in the AWS Well-Architected Tool (AWS WA Tool).

The ML Lens is now available as a custom lens for the AWS Well-Architected Tool in the AWS Management Console. Custom lenses, such as the ML Lens, are defined in a JSON file and allow you to tailor your workload reviews to particular technologies, help you meet governance needs, and extend the guidance already provided by the Well-Architected Framework and the AWS lenses.

To add the ML Lens to the AWS Well-Architected Tool:

1. Download the ML Lens JSON file. This file is used in Step 5.
2. Sign in to the AWS Management Console and open the AWS Well-Architected Tool console at https://console.aws.amazon.com/wellarchitected/.
3. In the left navigation pane, choose **Custom lenses**.
4. Choose **Create custom lens**.
5. Choose **Choose file** and select the JSON file you downloaded in Step 1.
6. (Optional) In the **Tags** section, add any tags you want to associate with the ML Lens.
7. Choose **Submit & Preview** to preview the ML Lens, or **Submit** to create the lens without previewing.

   If you choose to **Submit & Preview**, you can select **Next** to navigate through the ML Lens preview, or select **Exit Preview** to go back to **Custom lenses**.
8. Select the ML Lens and choose **Publish lens**.
9. In the **Version name** box, enter a unique identifier for the version change. This value can be up to 32 characters and must only contain alphanumeric characters and periods (".").

10. Choose **Publish custom lens**.

    After the ML Lens has been published, it's in **PUBLISHED** status.

The ML Lens can now be applied to workloads in your AWS account, and shared with other AWS accounts and users. If your account is managed by AWS Organizations, you can share the lens with all accounts in the organization or in an OU without having to enumerate each account.

As you work through the ML Lens checklist, risks can be identified and comments can be captured. A workload report is available in PDF format for sharing with stakeholders to document risks and future recommendations. Open risks can be managed and assigned in the tool and periodic milestone reviews can be performed.

For more information on using the AWS WA Tool, custom lenses, reports, and the risk dashboard, see the AWS Well-Architected Tool User Guide.

Well-Architected Framework pillars

The AWS Well-Architected Framework provides architectural best practices for designing and operating workloads in the cloud. The Framework consists of six pillars:

- **Operational excellence** - Includes the ability to run, monitor, and gain insights into workloads. It enables delivering business value and improves supporting processes and procedures. Best practice focus areas include: organization, prepare, operate, and evolve.
- **Security** - Includes the ability to protect information, systems, and assets. It enables delivering business value through risk assessments and mitigation strategies. Best practice focus areas include: security foundations, identity and access management, detection, infrastructure protection, data protection, incident response, and application security.
- **Reliability** - Includes the ability of a workload to recover from infrastructure or service disruptions. Ensures a workload performs its intended function correctly and consistently when it's expected to. It enables dynamically acquiring computing resources to meet demand, and mitigating disruptions such as misconfigurations and transient network issues. Best practice focus areas include: foundations, workload architecture, change management, and failure management.
- **Performance efficiency** - Focuses on the efficient use of computing resources to meet requirements. It enables maintaining efficiency as demand changes and technologies evolve. Best practice focus areas include: selection, review, monitoring, and trade-offs.
- **Cost optimization** - Includes the continuous process of refinement and improvement of a system over its entire lifecycle. It enables building and operating cost-aware systems that minimize costs, maximize return on investment, and achieve business outcomes. Best practice focus areas include: Cloud Financial Management, expenditure and usage awareness, resource cost-effectiveness, resource demand and supply management, and optimization.
- **Sustainability** - Focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. Best practice focus areas include: Region selection, alignment to demand, software and architecture, data, hardware and services, and process and culture.

While this paper focuses on the details specific to ML workloads, refer to the AWS Well-Architected Framework whitepaper for more information on the Framework and its pillars.

Well-Architected machine learning lifecycle

The ML lifecycle is the cyclic iterative process with instructions and best practices to use across defined phases while developing an ML workload. The ML lifecycle adds clarity and structure for making a machine learning project successful. The end-to-end machine learning lifecycle process illustrated in Figure 1 includes the following phases:

- Business goal identification
- ML problem framing
- Data processing (data collection, data preprocessing, feature engineering)
- Model development (training, tuning, evaluation)
- Model deployment (inference, prediction)
- Model monitoring

The phases of the ML lifecycle are not necessarily sequential in nature and can have feedback loops, a few of which are illustrated in Figure 1, to interrupt the cycle across the lifecycle phases.

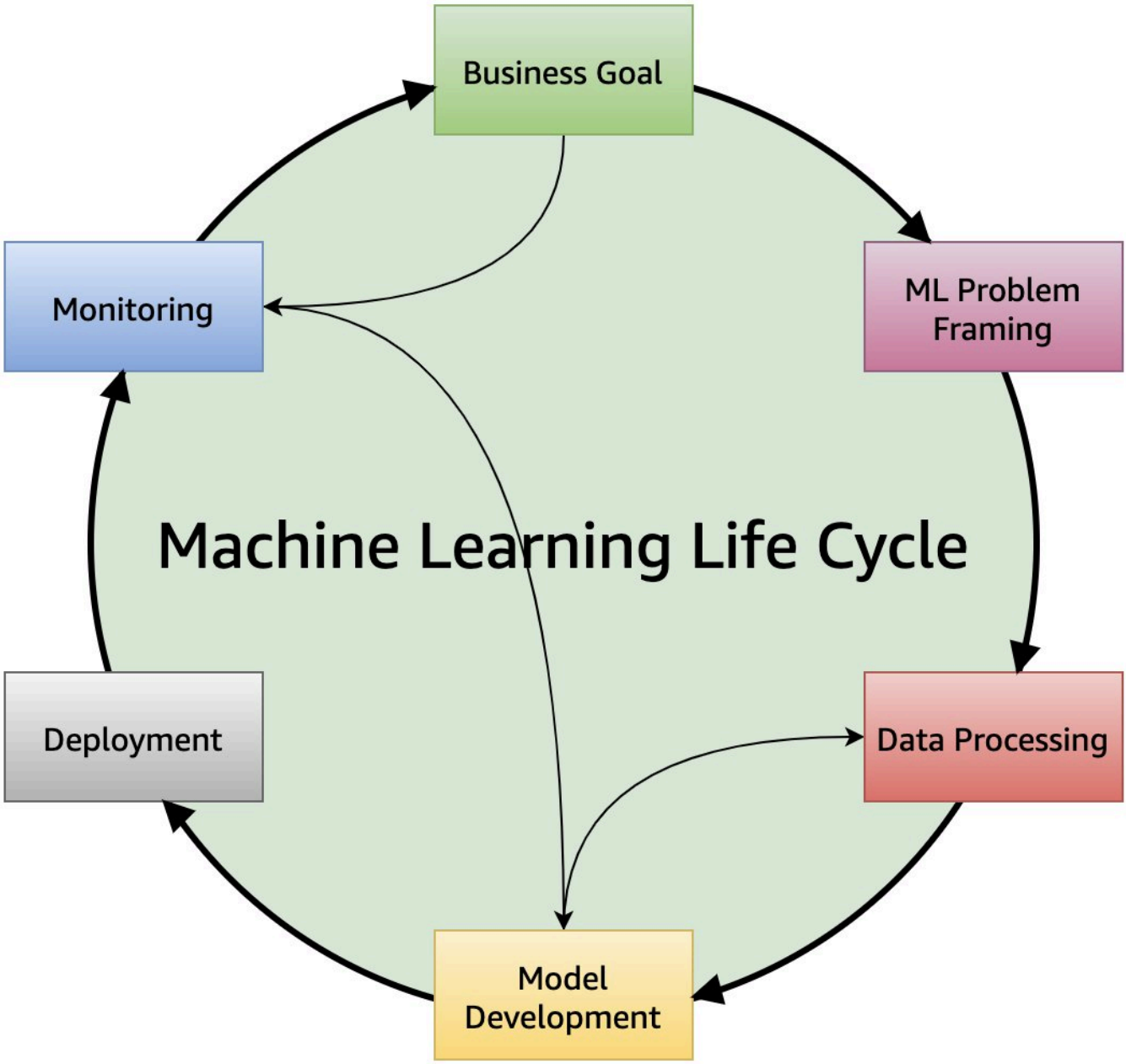

Figure 1: ML lifecycle

The following is a quick introduction to each phase, which will be expanded upon later in this paper.

Business goal

An organization considering ML should have a clear idea of the problem, and the business value to be gained by solving that problem. You must be able to measure business value against specific business objectives and success criteria.

ML problem framing

In this phase, the business problem is framed as a machine learning problem: what is observed and what should be predicted (known as a label or target variable). Determining what to predict and how performance and error metrics must be optimized is a key step in this phase.

Data processing

Training an accurate ML model requires data processing to convert data into a usable format. Data processing steps include collecting data, preparing data, and feature engineering that is the process of creating, transforming, extracting, and selecting variables from data.

Model development

Model development consists of model building, training, tuning, and evaluation. Model building includes creating a CI/CD pipeline that automates the build, train and release to staging and production environments.

Deployment

After a model is trained, tuned, evaluated and validated, you can deploy the model into production. You can then make predictions and inferences against the model.

Monitoring

Model monitoring system ensures your model is maintaining a desired level of performance through early detection and mitigation.

The Well-Architected ML lifecycle, shown in Figure 2, takes the machine learning lifecycle just described, and applies the Well-Architected Framework pillars to each of the lifecycle phases.

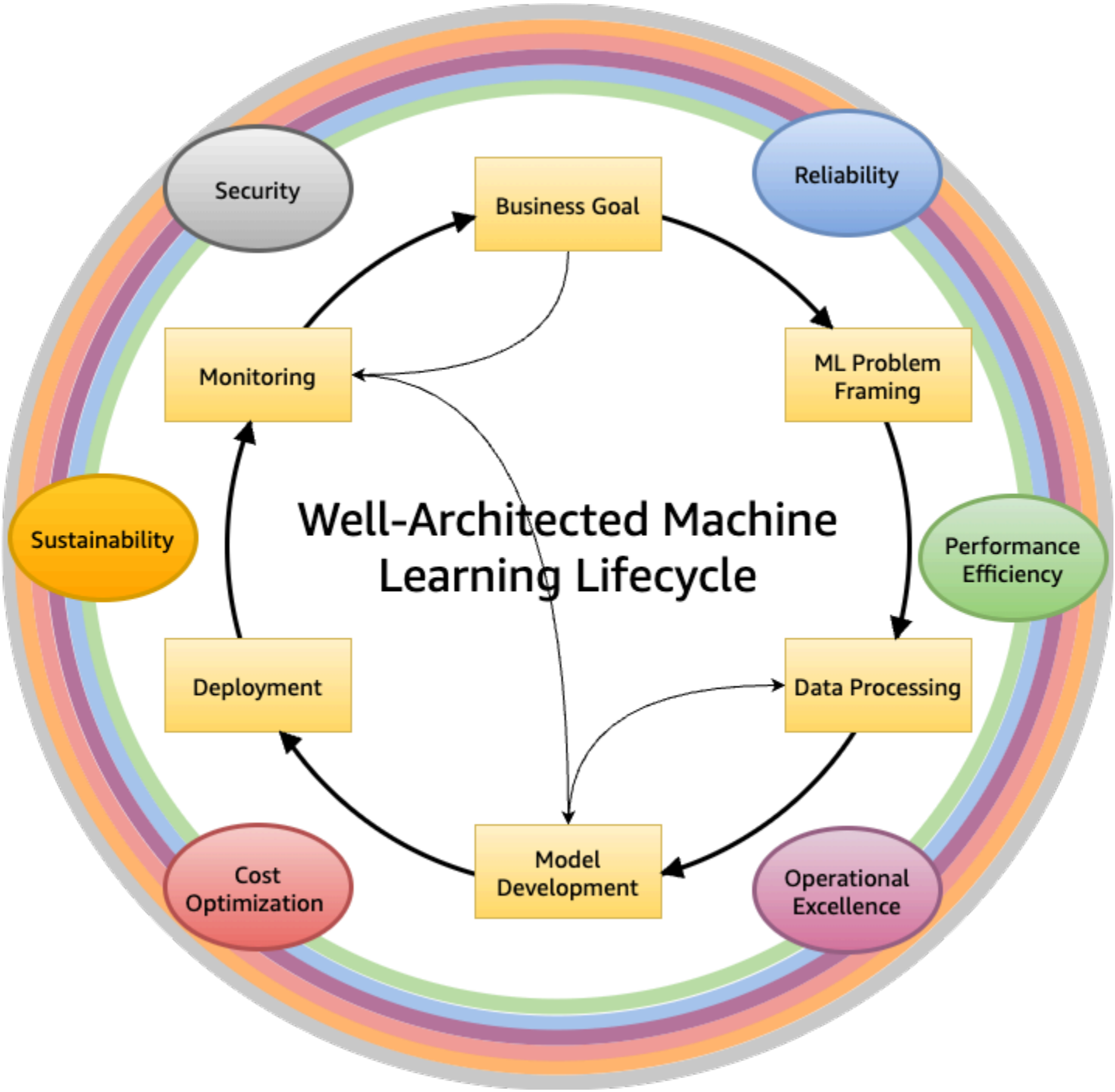

Figure 2: Well-Architected ML lifecycle

Well-Architected machine learning design principles

Well-Architected ML design principles are a set of considerations used as the basis for a well-architected ML workload.

Following the Well-Architected Framework guidelines, use these general design principles to facilitate good design in the cloud for ML workloads:

- **Assign ownership**- Apply the right skills and the right number of resources along with accountability and empowerment to increase productivity.
- **Provide protection** - Apply security controls to systems and services hosting model data, algorithms, computation, and endpoints. This ensures secure and uninterrupted operations.
- **Enable resiliency** - Ensure fault tolerance and the recoverability of ML models through version control, traceability, and explainability.
- **Enable reusability** - Use independent modular components that can be shared and reused. This helps enable reliability, improve productivity, and optimize cost.
- **Enable reproducibility** - Use version control across components, such as infrastructure, data, models, and code. Track changes back to a point-in-time release. This approach enables model governance and audit standards.
- **Optimize resources** - Perform trade-off analysis across available resources and configurations to achieve optimal outcome.
- **Reduce cost** - Identify the potentials for reducing cost through automation or optimization, analyzing processes, resources, and operations.
- **Enable automation** - Use technologies, such as pipelining, scripting, and continuous integration (CI), continuous delivery (CD), and continuous training (CT), to increase agility, improve performance, sustain resiliency, and reduce cost.
- **Enable continuous improvement** - Evolve and improve the workload through continuous monitoring, analysis, and learning.
- **Minimize environmental impact** - Establish sustainability goals and understand the impact of ML models. Use managed services and adopt efficient hardware and software and maximize their utilization.

Well-Architected machine learning

This section introduces ML specific Well-Architected best practices. For each of the ML lifecycle phases, Well-Architected best practices are examined across each of the six pillars of operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability. Best practices for each ML lifecycle phase follow an introductory background on each phase.

The six phases for the ML lifecycle referenced in this paper are illustrated in Figure 3 in a sequence.

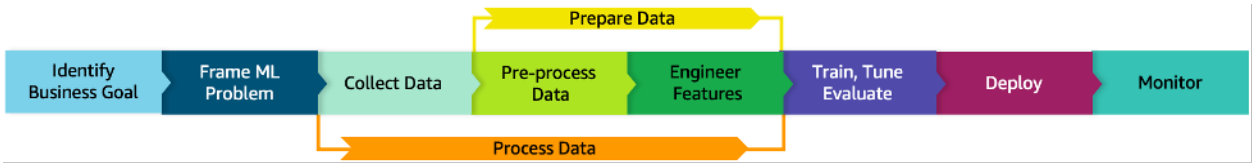

Figure 3: ML Lifecycle phases

The following sections describe the Well-Architected machine learning best practices for each of the lifecycle phases.

> **Note**
>
> When there is a best practice that applies to multiple pillars or phases, it is described in the pillar or phase where it makes the most impact. A complete list of the ML Lens best practices ordered by pillar instead of by ML lifecycle phase can be found in Best practices arranged by pillar.

Lifecycle phases

- ML lifecycle phase - Business goal
- ML lifecycle phase - ML problem framing
- ML lifecycle architecture diagram
- ML lifecycle phase - Data processing
- ML lifecycle phase - Model development
- ML lifecycle phase - Deployment
- ML lifecycle phase – Monitoring

ML lifecycle phase - Business goal

Business goal identification is the most important phase of the ML lifecycle. An organization considering ML should have a clear idea of the problem to be solved, and the business value to be gained. You must be able to measure business value against specific business objectives and success criteria. While this holds true for any technical solution, this step is particularly challenging when considering ML solutions because ML is a constantly evolving technology.

After you determine your criteria for success, evaluate your organization's ability to move toward that target. The target should be achievable and provide a clear path to production. Involve all relevant stakeholders from the beginning to align them to this target and any new business processes that result from this initiative.

Start the review by determining if ML is the appropriate approach for delivering your business goal. Evaluate all of the options that you have available for achieving the goal. Determine how accurate the resulting outcomes would be, while considering the cost and scalability of each approach.

For an ML-based approach to be successful, ensure that enough relevant, high-quality training data is available to the algorithm. Carefully evaluate the data to make sure that the correct data sources are available and accessible.

Steps in this phase:

The following work steps should be followed to establish your business goals.

- Business considerations
  - Understand business requirements.
  - Align affected stakeholders with this initiative.
  - Form a business question.
  - Identify critical, must-have features.
  - Consider new business processes that might come out of this implementation.
  - Consider how business value can be measured using business metrics that the ML model can help to improve.
- Frame the ML problem
  - Define the machine learning task based on the business question.
  - Review proven or published works in similar domains, if available.

    - Design small, focused POCs to validate those aspects of the approach where inadequate confidence exists.
- Determine the optimization objective
    - Determine key business performance metrics for the ML use case, such as uplift in new business acquisition, fraud detection rate, and anomaly detection. Increase CSAT according to the business needs.
- Review data requirements
    - Review the project's ML feasibility and data requirements.
- Cost and performance optimization
    - Evaluate the cost of data acquisition, training, inference, and wrong predictions.
    - Evaluate whether bringing in external data sources might improve model performance.
- Production considerations
    - Review how to handle ML-generated errors.
    - Establish pathways to production.

Best practices

- Operational excellence pillar - Best practices
- Security pillar - Best practices
- Reliability pillar - Best practices
- Performance efficiency pillar - Best practices
- Cost optimization pillar - Best practices
- Sustainability pillar - Best practices

Operational excellence pillar - Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures. This section includes best practices to consider while identifying the business goal.

Best practices

- MLOE-01: Develop the right skills with accountability and empowerment
- MLOE-02: Discuss and agree on the level of model explainability

- MLOE-03: Monitor model compliance to business requirements

MLOE-01: Develop the right skills with accountability and empowerment

Artificial intelligence (AI) has many different and growing branches, such as machine learning, deep learning, and computer vision. Given the complexity and fast-growing nature of ML technologies, plan to hire specialists with the understanding that additional training will be needed as ML evolves. Keep teams learning new skills, engaged, and motivated while encouraging accountability and empowerment at all times. Building ML models is a complex and iterative process that can infuse bias or unfair predictions against a certain entity. It's important to promote and enforce the ethical use of AI across enterprises. AWS provides clear guidance to customers for responsible AI practices.

Implementation plan

Develop skills - A key element in any organization's strategy for employee engagement and business growth must be ongoing learning and development. Consider strategies to grow your ML-driven business outcomes through intentional workforce skills development. Building a successful ML workforce includes providing training on ML concepts and algorithms, end-to-end ML lifecycle processes (such as model training, tuning, and deployment on Amazon SageMaker), and efficient use of ML infrastructure with SageMaker and automation with MLOps tools, such as SageMaker Pipelines. Training people in different specialty areas of ML, such as computer vision, NLP, and reinforcement learning based on your business needs, can increase productivity.

- **Develop accountability and empowerment** -AI applied through ML transforms the way business is run by tackling some of humanity's most challenging problems, augmenting human performance, and maximizing productivity. Promoting responsible use of these technologies is key to fostering continued innovation. Eliminating bias in datasets and model predictions by using Amazon SageMaker Clarify can help you build fair and explainable models.

Documents

- Responsible use of Artificial Intelligence and Machine Learning

Blogs

- Learn how SageMaker Clarify can detect bias

MLOE-02: Discuss and agree on the level of model explainability

Discuss and agree with the business stakeholders on the acceptable level of model explainability required for the use case. Use the agreed level as a metric for evaluations and tradeoff analysis across the ML lifecycle. Explainability can help with understanding the cause of a prediction, auditing, and meeting regulatory requirements. It can be useful for building trust ensuring that the model is working as expected.

Implementation plan

- **Understand business requirements** - The adoption of AI systems in regulated domains requires trust. This can be built by providing reliable explanations on the deployed predictions. Model explainability can be particularly important to reliability, safety, and compliance requirements. Use SageMaker Clarify to create explainability reports and detect dataset or model bias.
- **Agree on an acceptable level of explainability** - Communicate with stakeholders across the project about the level of explainability that is required for the project. Agree to a level that helps you meet business requirements.
- **Choose good baselines –** Shapley values determine the contribution that each feature made to model prediction. SHAP Baselines for Explainability are crucial to building fair and explainable ML models. Choose the baseline carefully since model explanations are based on deviations from the baseline (the baseline, in the ML context, is a hypothetical instance). You can choose a baseline with a 'low information content' (e.g., by constructing an average instance from the training dataset by taking either the median or average for numerical features and the mode for categorical features) or a baseline with 'high information content' (e.g., an instance which represents a particular class of instances that you are interested in). SageMaker Clarify, which uses Shapley Additive exPlanations (SHAP), calculates baselines automatically in the input dataset by using clustering methods such as K-means or K-prototypes. For more on SHAP baselines and parameters, please see the documents listed below.

Documents

- What Is Fairness and Model Explainability for Machine Learning Predictions?
- Model Explainability
- SHAP Baselines

MLOE-03: Monitor model compliance to business requirements

Machine learning models degrade over time due to changes in the real world, such as *data drift* and *concept drift*. If not monitored, these changes could lead to models becoming inaccurate or even obsolete over time. It's important to have a periodic monitoring process in place to make sure that your ML models continue to comply to your business requirements, and that deviations are captured and acted upon promptly.

Implementation plan

- **Agree on the metrics to monitor -** Clearly establish the metrics that you want to capture from your model monitoring process. These metrics should be tied to your business requirements and should cover your dataset-related statistics and model inference metrics.
- **Have an action plan on a drift –** If an unacceptable drift is detected in a dataset or the model output, have an action plan to mitigate it based on the type of drift and the metrics associated. This mitigation could include kicking off a retraining pipeline, updating the model, augmenting your dataset with more instances, or enriching your feature engineering process.

Documents

- Monitor models for data and model quality using SageMaker Model Monitor

Blogs

- Model monitoring at scale for production models

Security pillar - Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security posture. This section includes best practices to consider while identifying the business goal.

Best practices

- MLSEC-01: Validate ML data permissions, privacy, software, and license terms

MLSEC-01: Validate ML data permissions, privacy, software, and license terms

ML libraries and packages handle data processing, model development, training, and hosting. Establish a process to review the privacy and license agreements for all software and ML libraries needed throughout the ML lifecycle. Ensure these agreements comply with your organization's legal, privacy, and security terms and conditions. These terms should not add any limitations on your organization's business plans.

Implementation plan

- **Ensure data permissions for ML** - Verify whether the intended data can be used for machine learning, that it's a legitimate purpose, and whether you require additional consent from the data owner or data subjects. Have a plan to handle data subjects that subsequently withdraw their consent. Ensure documentation of data permissions is maintained for compliance purposes.
- **Bootstrap instances with lifecycle management policies** - Create a lifecycle configuration with a reference to your package repository, and a script to install required packages.
- **Evaluate package integrations that require external lookup services** - Based on your data privacy requirements, opt out of data collection when necessary. Minimize data exposure through trusted relationships. Evaluate the privacy policies and the license terms for ML packages that might collect data.
- **Use prebuilt containers** - Start with pre-packaged and verified containers to quickly provide support for commonly used dependencies. For example, AWS Deep Learning Containers contain several deep learning framework libraries and tools including TensorFlow, PyTorch, and Apache MXNet.

Documents

- Amazon Well-Architected Security Pillar for Software Integrity
- AWS Deep Learning Containers
- Installing External Libraries and Kernels on Notebook Instances
- AWS License Manager
- Lifecycle Configuration Best Practices

Blogs

- Private package installation in Amazon SageMaker running in internet-free mode

- Create a hosting VPC for PyPi package mirroring and consumption of approved packages
- Machine Learning Best Practices in Financial Services

Videos

- Machine Learning Best Practices in Financial Services

Reliability pillar - Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. There are no reliability best practices to consider while identifying the business goal.

Performance efficiency pillar - Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve. This section includes best practices to consider while identifying the business goal.

Best practices

- MLPER-01: Determine key performance indicators

MLPER-01: Determine key performance indicators

Use guidance from business stakeholders to capture key performance indicators (KPIs) relevant to the business use case. The KPIs should be directly linked to business value to guide acceptable model performance. Consider that machine learning inferences are probabilistic and will not provide exact results. Identify a minimum acceptable accuracy and maximum acceptable error in the KPIs. This helps enable achieving the required business value and manage the risk of variable results.

Implementation plan

- **Quantify the value of machine learning for the business** - Consider measures of how machine learning and automation will impact the business:
  - How much will machine learning reduce costs?
  - How many more users will be reached by increasing scale?

  - How much time will the business save by being able to respond faster to changes, such as in demand and supply disruptions?
  - How many hours of manual effort will be eliminated by automating with machine learning?
  - How much will machine learning be able to change user behavior, such as reducing churn?
- **Evaluate risks and the tolerance for error** -Quantify the impact of machine learning on the business. Rank order the value of impacts to identify the primary KPIs to optimize with machine learning. Define the cost of error for automated inferences that will be performed by ML models in the use case. Determine the tolerance of the business for error. For example, determine how far off a cost reduction estimate would have to be to negatively impact the business goals. Finally, evaluate the risks of machine learning for the business, and whether the benefits of ML solutions are of high enough value to outweigh those risks.

Documents

- Improve Business Outcomes with Machine Learning

Blogs

- Building the Business Case for Machine Learning in the Real World

Cost optimization pillar - Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment. This section includes best practices to consider while identifying the business goal.

Best practices

- MLCOST-01: Define overall return on investment (ROI) and opportunity cost
- MLCOST-02: Use managed services to reduce total cost of ownership (TCO)

MLCOST-01: Define overall return on investment (ROI) and opportunity cost

Evaluate the opportunity cost of ML for each use case to solve the business problem. Ensure cost effective decisions are made with respect to long-term resource allocation. Minimize the possible future risks and failures through upfront understanding of the ML development process and its resource requirements. Adopt automation and optimization that can result in reduced cost and improved performance.

Implementation plan

- **Specify the objectives of the ML project as research or development**. A research project is intended to discover the value that could be achieved from an untested ML use case, and the returns will be long-term. A development project is intended for specific production improvements, and is expected to deliver a faster return on investment. Both business management and data scientists should agree on whether the project is research-oriented, or development that applies well-understood methods to a well-known use case.
- **Use Tagging** so that costs can be tracked by project and business unit to give clear sight of ROI.
- **Evaluate and assess the data pipeline, the ML model, and the expected quality of production inferences** to estimate the costs of data and errors.
- **Develop a cost-benefit model**, and reassess that model as changes occur throughout the project. For example, changes in the external business environment, or the addition of expensive data sources, can require modifications to the initial cost-benefit model.
- **Understand, evaluate, and monitor** project risks.
- **Estimate the cost of resources needed,** such as data engineers and data scientists, to maintain a production model.

Documents

- Pricing for Amazon ML
- AWS Application Cost Profiler
- Managing costs with AWS Budgets
- AWS Cost Explorer
- AWS Cost and Usage Report

Blogs

- Building the Business Case for Machine Learning in the Real World

Videos

- APIs: ROI from artificial intelligence

MLCOST-02: Use managed services to reduce total cost of ownership (TCO)

Evaluate adopting managed services and pay-per-usage. Using managed services enables organizations to operate more efficiently with reduced resources and reduced cost.

Implementation plan

- **Use Amazon managed ML services** - Use Amazon SageMaker as a fully managed machine learning service to build, train, and deploy models at scale and at significantly lower costs. The total cost of ownership (TCO) of SageMaker over a three-year period is much lower than other self-managed cloud-based ML options, such as Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Elastic Kubernetes Service (Amazon EKS). SageMaker includes technologies such as Autopilot, Feature Store, Clarify, DataWrangler, Debugger, Studio, Training, Model deployment, Monitoring, and Pipelines.
- **Use Amazon managed AI services** - AWS pre-trained AI services provide ready-made intelligence for your applications and workflows. AI services integrate with your applications to address common use cases, such as personalized recommendations, modernizing your contact center, improving safety and security, and increasing customer engagement. AI services on AWS don't require machine learning experience. They are fully managed, complete solutions with pay-as-you-go pricing and no upfront commitment.
- **Perform pricing model analysis** - Analyze each component of the workload. Determine if the component and resources will be running for extended periods and eligible for commitment discounts, such as AWS Savings Plans. You can use Savings Plans to save on AWS usage in exchange for a commitment to a consistent amount of usage. Amazon SageMaker Savings Plans offer flexible attributes such as instance family, instance size, AWS Region, and component for your SageMaker instance usage.

Documents

- Amazon SageMaker Total Cost of Ownership (TCO)
- AWS Pricing Calculator for SageMaker
- Amazon managed AI services
- AWS Savings Plans

Blogs

- Lowering total cost of ownership for machine learning and increasing productivity with Amazon SageMaker
- Decrease Your Machine Learning Costs with Instance Price Reductions and Savings Plans for Amazon SageMaker
- Amazon SageMaker Continues to Lead the Way in Machine Learning and Announces up to 18% Lower Prices on GPU Instances

Videos

- Optimizing your Machine Learning costs on Amazon SageMaker with Savings Plans (April 2021)

Sustainability pillar - Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to guide direct action on how to reduce resource usage. This section includes best practices to consider while identifying the business goal.

Best practices

- MLSUS-01: Define the overall environmental impact or benefit

MLSUS-01: Define the overall environmental impact or benefit

Measure your workload's impact and its contribution to the overall sustainability goals of the organization.

Implementation plan

- **Ask probing questions** - How does this workload support our overall sustainability mission? How much data will we have to store and process? What is the impact of training the model? How often will we have to re-train? What are the impacts resulting from customer use of this workload? What will be the productive output compared with this total impact? Asking these questions will help you establish specific sustainability objectives and success criteria to measure against in the future.

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data
- New – Customer Carbon Footprint Tool

ML lifecycle phase - ML problem framing

In this phase, the business problem is framed as a machine learning problem: what is observed and what should be predicted (known as a label or target variable). Determining what to predict and how performance must be optimized is a key step in ML. For example, consider a scenario where a manufacturing company wants to maximize profits. There are several possible approaches including forecasting sales demand for existing product lines to optimize output, forecasting the required input materials and components required to reduce capital locked up in stock, and predicting sales for new products to prioritize new product development.

It's necessary to work through framing the ML problems in line with the business challenge.

Steps in this phase:

- Define criteria for a successful outcome of the project.
- Establish an observable and quantifiable performance metric for the project, such as accuracy.
- Define the relationship between the technical metric (for example, accuracy) and the business outcome (for example, sales).
- Help ensure business stakeholders understand and agree with the defined performance metrics.
- Formulate the ML question in terms of inputs, desired outputs, and the performance metric to be optimized.

- Evaluate whether ML is the right approach. Some business problems don't need ML as simple business rules can do a much better job. For other business problems, there might not be sufficient data to apply ML as a solution.
- Create a strategy to achieve the data sourcing and data annotation objective.
- Start with a simple model that is easy to interpret, and makes debugging more manageable.
- Map the technical outcome to a business outcome.
- Iterate on the model by gathering more data, optimizing the parameters, or increasing the complexity as needed to achieve the business outcome.

Best practices

- Operational excellence pillar - Best practices
- Security pillar - Best practices
- Reliability pillar - Best practices
- Performance efficiency pillar - Best practices
- Cost optimization pillar – Best practices
- Sustainability pillar - Best practices

Operational excellence pillar - Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures. This section includes best practices to consider while framing the ML problem.

Best practices

- MLOE-04: Establish ML roles and responsibilities
- MLOE-05: Prepare an ML profile template
- MLOE-06: Establish model improvement strategies
- MLOE-07: Establish a lineage tracker system
- MLOE-08: Establish feedback loops across ML lifecycle phases
- MLOE-09: Review fairness and explainability

MLOE-04: Establish ML roles and responsibilities

Understand the roles, responsibilities, ownership, and required interactions across teams to maximize overall effectiveness. An ML project typically consists of multiple roles, with defined tasks and responsibilities for each. In many cases, the separation of roles and responsibilities is not clear and there is overlap.

Implementation plan

- **Establish cross-functional teams with roles and responsibilities** - Enterprises often struggle getting started with their first enterprise-grade ML platform. This is partly due to the ML platform architecture having many components. Complexities around data science, data management, and model and operational governance also contribute to this struggle. Building an enterprise ML platform requires the collaboration of different cross-functional teams. The different personas from the different teams should each have a different role and responsibilities. They all should contribute in the build-out, usage, and operation of an ML platform. Structure your ML organization to support your defined business outcomes. Examples of ML technical roles and responsibilities with their definitions include:
  - **Domain expert** - Has valuable functional knowledge and understanding of the environment that the ML problem must be framed in. Helps ML engineers and data scientists with developing and validating assumptions and hypotheses. Engages early in the ML lifecycle and stays in close contact with the engineering owners throughout the evaluation phase.
  - **Data engineer** - Transforms data into a consumable format for machine learning and data science analysis.
  - **Data scientist** - Employs machine learning, statistical modeling, and artificial intelligence to derive insights from the data.
  - **ML engineer** - Turns reference implementations of ML models developed by data scientists into production-ready software.
  - **MLOps engineer** - Builds and manages automation pipelines to operationalize the ML platform and ML pipelines for fully or partially automated CI/CD pipelines. These pipelines automate building Docker images, model training, and model deployment. MLOps engineers also have a role in overall platform governance such as data and model lineage, as well as infrastructure and model monitoring.
  - **IT auditor** - Responsible for analyzing system access activities, identifying anomalies and violations, preparing audit reports for audit findings, and recommending remediations.

  - **Model risk manager** - Responsible for ensuring machine learning models meet various external and internal control requirements. These requirements include: model inventory, model explainability, model performance monitoring, and model lifecycle management.
  - **Cloud security engineer** - Responsible for creating, configuring, and managing the cloud accounts, and the resources in the accounts. Works with other security functions, such as identity and access management, to set up the required users, roles, and policies to grant users and services permissions to perform various operations in the cloud accounts. On the governance front, cloud security engineer implements governance controls such as resource tagging, audit trail, and other preventive and detective controls to meet both internal requirements and external regulations.
- **Enable easy mechanisms to control access and grant permissions for various ML roles** – Appropriate user access controls are essential for governance; they enable practitioners to access the tools they need to do their jobs, while ensuring data privacy and security.
  - Avoid the use of one-time methods for managing access policies for large teams which contain multiple roles for performing various ML activities, such as data preparation, training, and model monitoring.
  - Use Amazon SageMaker Role Manager to make it easier for administrators to control access and define permissions for users. Administrators can select and edit prebuilt templates based on various user roles and responsibilities. The tool then automatically creates the access policies with the necessary permissions within minutes, reducing the time and effort to onboard and manage users over time.

Documents

- Personas for an ML platform
- AWS MLOps Framework
- Why should you use MLOps?
- Amazon SageMaker Role Manager

Blogs

- Architecting Persona-centric Data Platform with on-premises Data Sources
- Define customized permissions in minutes with Amazon SageMaker Role Manager

- New ML Governance Tools for Amazon SageMaker – Simplify Access Control and Enhance Transparency Over Your ML Projects

MLOE-05: Prepare an ML profile template

Prepare an ML profile template to capture workload artifacts across ML lifecycle phases. The template helps enable evaluating the current maturity status of a workload and plan for improvements accordingly. Artifact examples to capture for the deployment phase include: model instance size, model update schedule, and model deployment location. This template should have artifact metrics with thresholds to evaluate and rank the level of maturity. Enable the ML profile template to reflect workload maturity status with snapshots of existing profiles, and alternative target profiles. Provide documentation with rationale for choosing one option over another that meets the business requirements.

Implementation plan

- **Capture ML workload deployment characteristics** - Capture the most impactful deployment characteristics of your ML workload. In this paper, we will highlight the characteristics as a sample profile template on AWS. The collected design and provisioning characteristics will help identify the optimal deployment architecture, including computing and inference instance types and sizes.
- **Map ML workload characteristics across a spectrum from lower to higher ranges** -Ideally, there should be at least two profile templates generated for each workload characteristic. One ML profile template gives a snapshot of the current workload profile. Another profile template can be instantiated to capture the target or future characteristics of the ML workload.

Documentation should provide the rationale for justifying the characteristic values in the target profile.

Sample design, architecture, and provisioning characteristics include:

- **Model deployment sample characteristics include:**
  - Model size (model.tar.gz) in bytes
  - Number of models deployed per endpoint
  - Instance size (for example, r5dn.4x.large) as suggested by the inference recommender
  - Retraining and model endpoint update frequency (hourly, daily, weekly, monthly, or per-event)
  - Model deployment location (on premises, Amazon EC2, container, serverless, or edge)

- **Architectural deployment sample characteristics** for the internal underlying algorithm or neural architecture includes:
  - Inference pipeline architecture (single endpoint, or chained endpoints)
  - Neural architecture (single framework (Scikit-learn), or multi-framework (PyTorch+ Scikit-learn + TensorFlow))
  - Containers (SageMaker prebuilt container, bring your own container)
  - Location of the containers and models (on premises, cloud, or hybrid)
  - Serverless inferencing (pay as you go)
- **Traffic pattern deployment sample characteristics include:**
  - Traffic pattern (steady, or spiky)
  - Input size (number of bytes)
  - Latency (low, medium, high, or batch)
  - Concurrency (single thread, or multi-thread)
- **Cold start tolerance characteristics** - Determine and document the tolerance of the various aspects of cold start in milliseconds.
- **Network deployment characteristics** - Check for the applicability of network deployment characteristics including AWS KMS encryption, multi-variant endpoints, network isolation, and third-party Docker repositories.
- **Cost considerations** - Discuss and document the cost considerations for elements, such as Amazon EC2 Spot Instances.
- **Determine provisioning matrix** - Critical ML workloads might be vying for resources from cloud providers. For staging and production environments, include a matrix of the expected capacity requirements. This matrix consists of the number of instance types per AWS Region across training, batch interference, real-time inference, and notebooks.

MLOE-06: Establish model improvement strategies

Plan improvement drivers for optimizing model performance before ML model development starts. Examples of improvement drivers include: collecting more data, cross-validation, feature engineering, tuning hyperparameters, and ensemble methods.

Implementation plan

- **Use Amazon SageMaker Experiments** - Improvement strategies for ML experimentation follow a sequence from simple to more complex. Begin with minimal data cleaning and the most obvious data. Train a simple classical model using algorithms, such as linear regression or logistic regression, for classification tasks. Iterate by increasing the data processing and model complexity to improve metrics related to business value. Amazon SageMaker Experiments can help to organize multiple tests to compare different configurations and algorithms. A few sample approaches to experiment with include:

- **Use effective feature selection** - Work with subject matter experts to gain insight into the most significant features that will be related to the target values. Iteratively add more complex features, and remove less important features to improve model accuracy and robustness.
- **Use deep learning** - For a large volume training data, consider deep learning models to find previously unknown features and improve the model accuracy.
- **Consider ensemble methods** - Ensemble methods can add further improvements to accuracy by combining the best characteristics of various algorithms. However, there is a trade-off with computational performance and maintenance difficulty that should be considered for each specific business use case.
- **Consider AutoML** - Automatic machine learning, known as AutoML, removes the tedious, iterative, and time-consuming work across the machine learning workflow from data acquisition to model operationalization, so you can spend less time on low level details and more time on using ML to improve business outcomes. AutoML tools take care of sourcing and preparing data, engineering features, training and tuning models, deploying models, and ongoing model monitoring and updating. Amazon SageMaker Autopilot is an AutoML solution for tabular data.

- **Optimize hyperparameters** - Optimize hyperparameters for each algorithm to obtain the top performance before selection of the most appropriate. Amazon SageMaker Hyperparameter Optimization automates this process of selecting the top performance.

Documents

- Manage Machine Learning with Amazon SageMaker Experiments
- Perform Automatic Model Tuning with SageMaker

- Automate the experimental trials into SageMaker Pipelines by leveraging the native SageMaker Pipelines integration with experiments
- Automate model development with Amazon SageMaker Autopilot

Blogs

- Developing a business strategy by combining machine learning with sensitivity analysis
- Amazon SageMaker Experiments – Organize, Track And Compare Your Machine Learning Trainings
- Code-free machine learning: AutoML with AutoGluon, Amazon SageMaker, and AWS Lambda

Videos

- Hyperparameter Tuning with Amazon SageMaker's Automatic Model Tuning

Examples

- Ensemble Predictions from Multiple Models

MLOE-07: Establish a lineage tracker system

Maintain a system that tracks changes for each release. These changes include documentation, environment, model, data, code, and infrastructure. Having this system allows you to go back and quickly reproduce a problem on a prior release, allowing rollbacks and reproducibility.

Implementation plan

- **Identify artifacts needed for tracking** - Tracking all the artifacts used for a production model is an essential requirement for reproducing the model to meet regulatory and control requirements. Data and artifacts lineage tracking includes the list of artifacts needed for tracking.
- **Use SageMakerML Lineage Tracking** - SageMaker ML Lineage Tracking creates and stores information about the steps of an ML workflow from data preparation to model deployment. With the tracking information, you can reproduce the workflow steps, track model and data set lineage, and establish model governance and audit standards.
- **Use SageMaker Studio** - Use SageMaker Studio to track the lineage of a SageMaker ML pipeline.

- **Use SageMaker Feature Store** – Amazon SageMaker Feature Store is a purpose-built repository where you can store and access features so it's much easier to name, organize, and reuse them across teams. SageMaker Feature Store provides a unified store for features during training and real-time inference without the need to write additional code or create manual processes to keep features consistent
- **Use SageMaker Model Registry** - Use SageMaker Model Registry to catalog models for production, manage model versions, and associate metadata with a model. Model Registry enables lineage tracking.
- **Use SageMaker Pipelines for model building** -With SageMaker Pipelines you can track the history of your data within the pipeline. SageMaker ML Lineage Tracking lets you analyze input data, its source, and the outputs generated.

Documents

- SageMaker ML Lineage Tracking
- Data and artifacts lineage tracking
- SageMaker Model Building Pipelines
- Track the Lineage of a SageMaker ML Pipeline
- SageMaker Studio
- SageMaker Model Registry
- SageMaker Feature Store

Blogs

- Using model attributes to track your training runs on Amazon SageMaker

Examples

- Controlling and auditing data exploration activities with SageMakerStudio and AWS Lake Formation

MLOE-08: Establish feedback loops across ML lifecycle phases

Establish a feedback mechanism to share and communicate successful development experiments, analysis of failures, and operational activities. This facilitates continuous improvement on future

iterations of the ML workload. ML feedback loops are driven by model drifts and requires ML practitioners to analyze and revisit monitoring and retraining strategies over time. ML feedback loops allow experimentation with data augmentation, and different algorithms and training approaches until an optimal outcome is achieved. Document your findings to identify key learnings and improve processes over time.

Implementation plan

- **Establish SageMaker Model Monitoring** - The accuracy of ML models can deteriorate over time, a phenomenon known as model drift. Many factors can cause model drift, such as changes in model features. The accuracy of ML models can also be affected by concept drift, the difference between data used to train models and data used during inference. Amazon SageMaker Model Monitor continually monitors machine learning models for concept drift and model drift. SageMaker Model Monitor alerts you if there are any deviations so that you can take remedial action.
  - **Use Amazon CloudWatch** - Configure Amazon CloudWatch to receive notifications if a drift in model quality is observed. Monitoring jobs can be scheduled to run at a regular cadence (for example, hourly or daily) and push reports as well as metrics to Amazon CloudWatch and Amazon S3.
  - **Use Amazon SageMaker Model Dashboard** as the central interface to track models, monitor performance, and review historical behavior
  - **Automate retraining pipelines** - Create a CloudWatch Events rule that alerts on a events emitted by the SageMaker Model Monitoring system. The event rule can detect the drifts or anomalies, and start a retraining pipeline.
- **Use Amazon Augmented AI (A2I)** - Check accuracy by having human reviews to establish the *ground truth*, using tools such as Amazon A2I, against which model performance can be compared.

Documents

- Amazon SageMaker Model Monitor
- Creating a CloudWatch Events Rule That Triggers on an Event
- SageMaker Model Dashboard
- Monitoring Amazon ML with Amazon CloudWatch Metrics

Blogs

- Automated monitoring of your machine learning models with Amazon SageMakerModel Monitor and sending predictions to human review workflows using Amazon A2I
- Automating model retraining and deployment using the AWS Step Functions Data Science SDK for Amazon SageMaker
- Monitoring in-production ML models at large scale using Amazon SageMaker Model Monitor
- Human-in-the-loop review of model explanations with Amazon SageMaker Clarify and Amazon A2I
- Amazon SageMaker Model Monitor now supports new capabilities to maintain model quality in production

Videos

- Easily Implement Human in the Loop into Your Machine Learning Predictions with Amazon A2I

MLOE-09: Review fairness and explainability

Consider fairness and explainability during each stage of the ML lifecycle. Compile a list of questions to review for each stage including:

- **Problem framing** - Is an algorithm an ethical solution to the problem?
- **Data management** - Is the training data representative of different groups? Are there biases in labels or features? Does the data need to be modified to mitigate bias?
- **Training and evaluation** - Do fairness constraints need to be included in the objective function? Does changing the number of models to train needed to mitigate bias? Has the model been evaluated using relevant fairness metrics?
- **Deployment** - Is the model deployed on a population for which it was not trained or evaluated?
- **Monitoring** - Are there unequal effects across users?

Implementation plan

- **Use Amazon SageMaker Clarify** - Understand model characteristics, debug predictions, and explain how ML models make predictions with Amazon SageMaker Clarify. Amazon SageMaker Clarify uses a model-agnostic feature attribution approach that includes an efficient implementation of SHAP (Shapley Additive Explanations). SageMaker Clarify allows you to:

- Understand the compliance requirements for fairness and explainability.
- Determine whether training data is biased in its classes or population segments, particularly protected groups.
- Develop a strategy for monitoring for bias in data when the model is in production.
- Consider the trade-offs between model complexity and explainability, and select simpler models if explainability is required.

Documents

- What Is Fairness and Model Explainability for Machine Learning Predictions?
- Amazon SageMaker Clarify: Detect bias in ML models and understand model predictions
- Feature Attributions that Use Shapley Values
- Amazon SageMaker Clarify: Machine Learning Bias Detection and Explainability in the Cloud

Blogs

- ML model explainability with Amazon SageMaker Clarify and the SK Learn pre-built container
- Explaining Amazon SageMaker Autopilot models with SHAP

Videos

- Machine learning and society: Bias, fairness, and explainability
- How Clarify helps machine learning developers detect unintended bias
- Interpretability and explainability in machine learning

Security pillar - Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. This section includes best practices to consider while framing the ML problem.

Best practices

- MLSEC-02: Design data encryption and obfuscation

MLSEC-02: Design data encryption and obfuscation

Consider how personal data should be protected. Field level encryption or obfuscation can be used to protect personally identifiable data.

Implementation plan

- **Audit data for attributes requiring special treatment** - Identify fields containing data requiring special treatment, such as field level encryption, data masking or obfuscation

Blogs

- Introducing PII Data Identification and Handling Using AWS Glue Databrew

- 7 ways to improve security of your machine learning workflows
- Secure deployment of Amazon SageMaker resources

Reliability pillar - Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This section includes best practices to consider while framing the ML problem.

Best practices

- MLREL-01: Use APIs to abstract change from model consuming applications
- MLREL-02: Adopt a machine learning microservice strategy

MLREL-01: Use APIs to abstract change from model consuming applications

Use a flexible application and API design to abstract change from model consuming applications. Ensure that changes to an ML model are introduced with minimal or no interruption to existing workload capabilities. Minimize the changes across other downstream applications.

Implementation plan

- **Adopt best practices in use of APIs** -Expose your ML endpoints through APIs so that changes to the model can be introduced without disrupting upstream communications. Document your API

in a central repository or documentation site so that any calling services can easily understand your API routes and flags. Ensure that any changes to your API are communicated with any calling services.

- **Deploy a model in Amazon SageMaker**- After you train your model, you can deploy it using Amazon SageMaker to get predictions. To establish a persistent endpoint to get one prediction at a time, use SageMaker hosting services. To get predictions for an entire dataset, use SageMaker batch transform.
- **Use Amazon API Gateway to create APIs** - Amazon API Gateway is a fully managed service that enables developers to create, publish, maintain, monitor, and secure APIs. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.

Documents

- Deploy a Model in Amazon SageMaker
- What is Amazon API Gateway?
- API Gateway Pattern

Blogs

- Build a serverless frontend for an Amazon SageMaker endpoint
- Creating a machine learning-powered REST API with Amazon API Gateway mapping templates and Amazon SageMaker
- Deploying machine learning models with serverless templates

Videos

- Deploy Your ML Models to Production at Scale with Amazon SageMaker

Examples

- AWS Solutions Constructs aws-apigateway-sagemaker endpoint
- AWS MLOps Framework
- Amazon SageMaker Safe Deployment Pipeline

- Amazon SageMaker Inference Client Application

MLREL-02: Adopt a machine learning microservice strategy

Where appropriate, a complex business problem can be usefully decomposed into a series of machine learning models with a loosely coupled implementation. This can be accomplished by adopting a microservice instead of a monolithic architecture. This approach replaces one large resource with multiple small resources and can reduce the impact of a single failure on the overall workload. This strategy enables distributed development and improves scalability, enabling easier change management.

Implementation plan

- **Adopt a microservice strategy** - Service-oriented architecture (SOA) is the practice of making software components reusable using service interfaces. Instead of building a monolithic application, where all functionality is contained in a single runtime, the application is instead broken into separate components. Microservices extend this by making components that are single-purpose and reusable. When building your architecture, divide components along business boundaries or logical domains. Adopt a philosophy that favors single-purpose applications that can be composed in different ways to deliver different end-user experiences.
- **Use AWS services in developing microservices** - Two popular approaches for developing microservices are using AWS Lambda and Docker containers with AWS Fargate. With AWS Lambda, you can run code for virtually any type of application or backend service with zero administration. You pay only for the compute time you consume, and there is no charge when your code is not running. A common approach to reduce operational efforts for deployment is using a container- based deployment. AWS Fargate is a container management service that allows you to run serverless containers so you don't have to worry about provisioning, configuring, and scaling clusters of virtual machines to run containers.

Documents

- Implementing Microservices on AWS
- Microservices on AWS
- Break a Monolith Application into Microservices
- AWS Lambda Documentation
- What is AWS Fargate?

Blogs

- Deploying Python Flask microservices to AWS using open-source tools
- Deploying machine learning models as serverless APIs
- Integrating machine learning models into your Java-based microservices
- Adopting machine learning in your microservices with DJL (Deep Java Library) and Spring Boot
- Building, deploying, and operating containerized applications with AWS Fargate

Videos

- Breaking the Monolith Using AWS Container Services
- AWS New York Summit 2019: Migrating Monolithic Applications with the Strangler Pattern (FSV303)

Examples

- Run a Serverless “Hello, World” with AWS Lambda

Performance efficiency pillar - Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve. This section includes best practices to consider while framing the ML problem.

Best practices

- MLPER-02: Use purpose-built AI and ML services and resources
- MLPER-03: Define relevant evaluation metrics

MLPER-02: Use purpose-built AI and ML services and resources

Consider how part or all of the workload could be handled by pre-built AI services or ML resources. Better performance can often be delivered more efficiently by using pre-optimized components included in AI and ML managed services. Select an optimal mix of bespoke and pre-built components to meet the workload requirements.

Implementation plan

- **Learn about AWS AI services** - Determine whether AWS managed AI services are applicable to the business use case. Understand how managed AWS AI services can relieve the burden of training and maintaining an ML pipeline. Use Amazon SageMaker to develop in the cloud and understand the roles and responsibilities needed to maintain the ML workload. Consider combining managed AI services with custom ML models built on Amazon SageMaker. This approach allows balancing the tradeoffs between ML workload management, and solutions specificity for the business use case.
- **Learn about** SageMaker JumpStart – This service provides pre-trained, open-source models for a wide range of problem types to help you get started with machine learning.
- **Learn about** SageMaker Algorithms and Models in AWS Marketplace, a curated digital catalog that makes it easy for you to find, buy, deploy, and manage third-party software and services that can help you build solutions and run their businesses.

Documents

- Overview of Amazon Web Services: Machine Learning
- Machine Learning on AWS
- Architecture Best Practices for Machine Learning

Videos

- An Overview of AI and Machine Learning Services From AWS

MLPER-03: Define relevant evaluation metrics

To validate and monitor model performance, establish numerical metrics that directly relate to the KPIs. These KPIs are established in the business goal identification phase. Evaluate whether the performance metrics accurately reflect the business' tolerance for the error. For instance, false positives might lead to excessive maintenance costs in predictive maintenance use cases. Numerical metrics, such as precision and recall, would help differentiate the business requirements and be closer aligned to business value. Consider developing custom metrics that tune the model directly for the business objectives. Examples of standard metrics for ML models include:

- Classification

  - Confusion matrix (precision, recall, accuracy, F1 score)
  - Receiver operating characteristic-area under curve (AUC)
  - Logarithmic loss (log-loss)
- Regression
  - Root mean square error (RMSE)
  - Mean absolute percentage error (MAPE)

Implementation plan

- **Optimize business-related metrics** - Identify performance metrics relevant to use-case and model type. Implement the metric as a loss function or use the loss function included in Amazon SageMaker. Use Amazon SageMaker Experiments to evaluate the metrics with consideration to the business use case to maximize business value. Track model and concept drift in real time with Amazon SageMaker Model Monitor to estimate errors.
  - Calculate the maximum probability of error that will be required for the ML model to produce results considering the tolerance set by the business.
  - Select and train ML models on the available data to make prediction within the probability bounds. Organize tests on different models with Amazon SageMaker Experiments.

Documents

- Monitor and Analyze Training Jobs Using Metrics
- Manage Machine Learning with Amazon SageMaker Experiments

Blogs

- Training models with unequal economic error costs using Amazon SageMaker
- Amazon SageMaker Experiments – Organize, Track, and Compare Your Machine Learning Trainings

Videos

- Organize, Track, and Evaluate ML Training Runs with Amazon SageMaker Experiments

Examples

- Scikit-Learn Data Processing and Model Evaluation

Cost optimization pillar – Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment. This section includes best practices to consider while framing the ML problem.

Best practices

- MLCOST-03: Identify if machine learning is the right solution
- MLCOST-04: Tradeoff analysis on custom versus pre-trained models

MLCOST-03: Identify if machine learning is the right solution

Evaluate if there are alternatives, such as a simple rule-based approach, that could do a better job than ML. Weigh the cost of adopting ML against the opportunity cost of not leaning on ML transformation. Specialized resources, such as data scientist time or model time-to-market, might be the most expensive and constrained resources. The most cost-effective hardware choice might not be cost optimized if it constrains experimentation and development speed.

Implementation plan

- Start simple:
  - Articulate your problem.
  - Identify your data sources.
  - Think about cost involved in:
    - Designing or preparing your data for the model.
    - Data storage cost for ML.
    - Model training cost depending on the hardware. choice
    - Data labeling cost, if required.
    - Potential bias resulting in iterative model re-training leading to higher cost.

    - Potential cost of hosting the ML model.
    - Model maintenance costs.
  - Consider these data points to weigh the cost of adopting ML against the opportunity cost of not leaning on ML transformation.
- Use Amazon SageMaker Autopilot and SageMaker Clarify to validate that ML is the right solution.
  - **Baseline the solution** by reviewing how the problem is solved today. If a rules-based solution is available, then use it as a baseline. Selecting a simple ML model for baselining can also be done using JumpStart or AWS Marketplace. AWS also provides many pre-built solutions with one-click deploy for most common business use cases.
  - **Build a machine learning model** using SageMaker or SageMaker Autopilot and compare the metrics of this solution against the baseline.
  - **Use** SageMaker Clarify to explain the model that you have built using SageMaker or Autopilot.
  - **Identify** if the ML model is performing better than your existing solution or a rules-based approach before investing on an ML-based solution.

Documents

- Amazon SageMaker
- Amazon SageMaker Autopilot
- Amazon SageMaker Jumpstart
- Amazon SageMaker Clarify
- Machine Learning solutions in AWS Marketplace

MLCOST-04: Tradeoff analysis on custom versus pre-trained models

Optimize the cost through tradeoff analysis based on custom versus pre-trained models. This tradeoff analysis should keep the security and performance efficiency in perspective and within the acceptable thresholds.

Implementation plan

- **Use Amazon SageMaker built-in algorithms and AWS Marketplace** - Amazon SageMaker provides a suite of built-in algorithms to help data scientists and machine learning practitioners get started on training and deploying machine learning models. Pre-trained ML models are

ready-to-use models that can be quickly deployed on Amazon SageMaker. By pre-training the ML models for you, solutions in the AWS Marketplace take care of the heavy lifting, helping you deliver AI- and ML-powered features faster and at a lower cost. Evaluate the cost of your data scientists' time and other resource requirements to develop your own custom model vs. bringing a pre-trained model and deploying it on SageMaker for inferencing. The advantage of a custom model is the flexibility to fine-tune it to match the needs of your business use case. A pre-trained model can be difficult to modify and you might have to use it as is.

- **Use Amazon SageMaker Jumpstart** to access pre-trained models and accelerate the ML development process. SageMaker JumpStart provides a set of solutions for the most common use cases that can be deployed readily with just a few clicks. The solutions are fully customizable and showcase the use of AWS CloudFormation templates and reference architectures so you can accelerate your ML journey. Amazon SageMaker JumpStart also supports one-click deployment and fine-tuning of more than 150 popular open-source models such as natural language processing, object detection, and image classification models.

Documents

- Pre-trained machine learning models available in AWS Marketplace
- Amazon SageMaker Jumpstart

Blogs

- Bring your own pre-trained MXNet or TensorFlow models into Amazon SageMaker
- How Startups Deploy Pretrained Models on Amazon SageMaker
- Amazon SageMaker JumpStart Simplifies Access to Pre-built Models and Machine Learning Solutions
- Amazon SageMaker JumpStart models and algorithms now available via API
- Machine Learning algorithms and model packages now available in AWS Marketplace
- Using Amazon Augmented AI with AWS Marketplace machine learning models
- Save costs by automatically shutting down idle resources within Amazon SageMaker Studio

Sustainability pillar - Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. This section includes best practices to consider while framing the ML problem.

Best practices

- MLSUS-02: Consider AI services and pre-trained models
- MLSUS-03: Select sustainable Regions

Related best practices

- **Identify if machine learning is the right solution** (MLCOST-03) - Always ask if AI or ML is right for your workload. There is no need to use computationally intensive AI when a simpler, more sustainable approach might be just as successful. For example, using ML to route Internet of Things (IoT) messages might be unwarranted. Instead, you might be able to express the logic with a rules engine.

MLSUS-02: Consider AI services and pre-trained models

Consider whether the workload needs to be developed as a custom model. Many workloads can use managed AI services accessible through an API. Using these services means that you won't need to provision your own resources to collect, store, and process training data and to prepare, train, tune, and deploy an ML model.

If adopting a fully managed AI service is not appropriate, evaluate if you can use pre-existing datasets, algorithms, or models. You can also fine-tune an existing model starting from a pre-trained model. Using pre-trained models from third parties can reduce the resources needed for data preparation and model training.

Implementation plan

- **Use pre-trained AWS AI services** - AWS AI services integrate with applications through APIs to address common use cases such as personalized recommendations, image recognition, language analysis and translation, modernizing contact centers, improving safety and security, and increasing customer engagement.
- **Use pre-trained models from AWS Marketplace** - AWS Marketplace offers over 1,400 ML-related assets that you can subscribe to.

- **Use pre-trained models from SageMaker JumpStart** - SageMaker JumpStart provides pre-trained, open-source models for a wide range of problem types to help you get started with machine learning. You can incrementally train and tune these models before deployment.

Documents

- Explore AWS AI services
- Pre-trained machine learning models available in AWS Marketplace
- Use Hugging Face with Amazon SageMaker
- Use SageMaker JumpStart algorithms with pre-trained models

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data
- Fine-tune and host Hugging Face BERT models on Amazon SageMaker

Videos

- Introduction to Hugging Face on Amazon SageMaker

MLSUS-03: Select sustainable Regions

Choose the Regions where you implement your workloads based on both your business requirements and sustainability goals.

Implementation plan

- **Select an AWS Region with sustainable energy sources** - When regulations and legal aspects allow, choose Regions near Amazon renewable energy projects and Regions where the grid has low published carbon intensity to host your data and workloads.

Resources

- Renewable energy projects on Amazon Around the Globe
- Renewable Energy Methodology

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data
- How to select a region for your workload based on sustainability goals

ML lifecycle architecture diagram

Figure 4 shows the ML lifecycle phases with the “data processing phase” (for example, “Process Data”) expanded into a “data collection sub-phase” (“Collect Data”) and a “data preparation sub-phase” (“Pre-process Data” and “Engineer Features”). These sub-phases are discussed in more detail in this section.

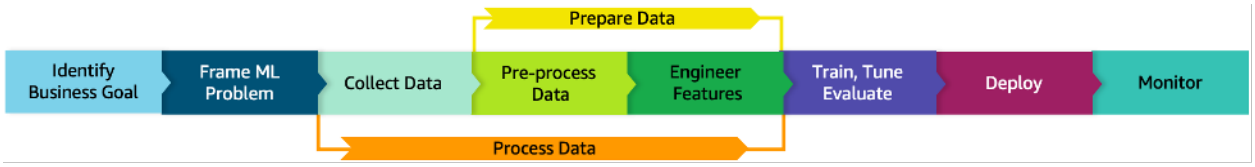

Figure 4: ML lifecycle with data processing sub-phases included

Figure 5 illustrates the details of all the ML lifecycle phases that occur following the problem framing phase and shows how the data-processing sub-phases interact with the subsequent phases, that is, the “model development phase”, the “model monitoring phase”, and the “model monitoring phase”.

The model development phase includes training, tuning, and evaluation. The model deployment phase includes the staging environment for model validation for security and robustness. Monitoring is key in timely detection and mitigation of drifts. Feedback loops across the ML lifecycle phases are key enablers for monitoring. Feature stores (both online and offline) provide consistent and reusable features across model development and deployment phases. The model registry enables the version control and lineage tracking for model and data components. This figure also emphasizes the lineage tracking and its components that are discussed in this section in more detail.

The cloud agnostic architecture diagrams in this paper provide high-level best practices with the following assumptions:

- All concepts presented here are cloud and technology agnostic.
- Solid black lines are indicative of process flow.

- Dashed color lines are indicative of input and output flow.
- Architecture diagram components are color-coded for ease of communication across this document.

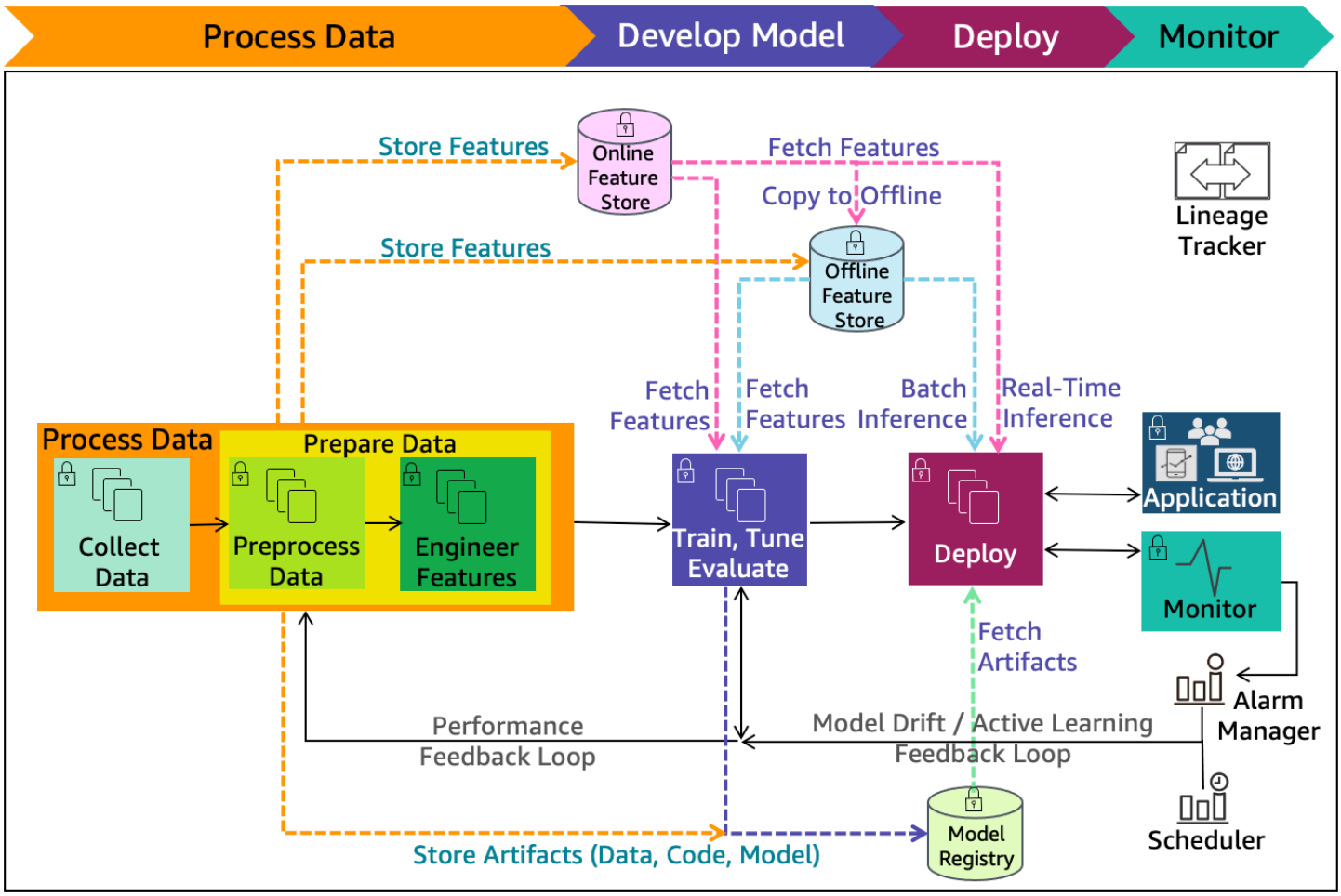

Figure 5: ML lifecycle with detailed phases and expanded components

The components of the sub-phases of the ML lifecycle shown in Figure 5 are as follows:

- **Online/Offline feature store** - Reduces duplication and the need to rerun feature engineering code across teams and projects. An online store with low-latency retrieval capabilities is ideal for real-time inference. On the other hand, an offline store is designed for maintaining a history of feature values and is suited for training and batch scoring.
- **Model registry** - A repository for storing ML model artifacts including trained model and related metadata (such as data, code, and model). It enables the tracking of the lineage of ML models as it can act as a version control system.

- **Performance feedback loop** - Informs the iterative data preparation phase based on the evaluation of the model during the model development phase.
- **Model drift feedback loop** - Informs the iterative data preparation phase based on the evaluation of the model during the production deployment phase.
- **Alarm manager** - Receives alerts from the model monitoring system. It then publishes notifications to the services that can deliver alerts to target applications. The model update re-training pipeline is one such target application.
- **Scheduler** - Initiates a model re-training at business-defined intervals.
- **Lineage tracker** - Enables reproducible machine learning experiences. It enables the re-creation of the ML environment at a specific point in time, reflecting the versions of all resources and environments at that time.

The ML lineage tracker collects references to traceable data, model, and infrastructure resource changes. It consists of the following components:

- System architecture (infrastructure as code to address environment drift)
- Data (metadata, values, and features)
- Model (algorithm, features, parameters, and hyperparameters)
- Code (implementation, modeling, and pipeline)

The lineage tracker collects changed references through alternative iterations of ML lifecycle phases. Alternative algorithms and feature lists are evaluated as experiments for final production deployment.

Figure 6 includes machine learning components and their information that the lineage tracker collects across different releases. The collected information enables going back to a specific point-in-time release and re-creating it.

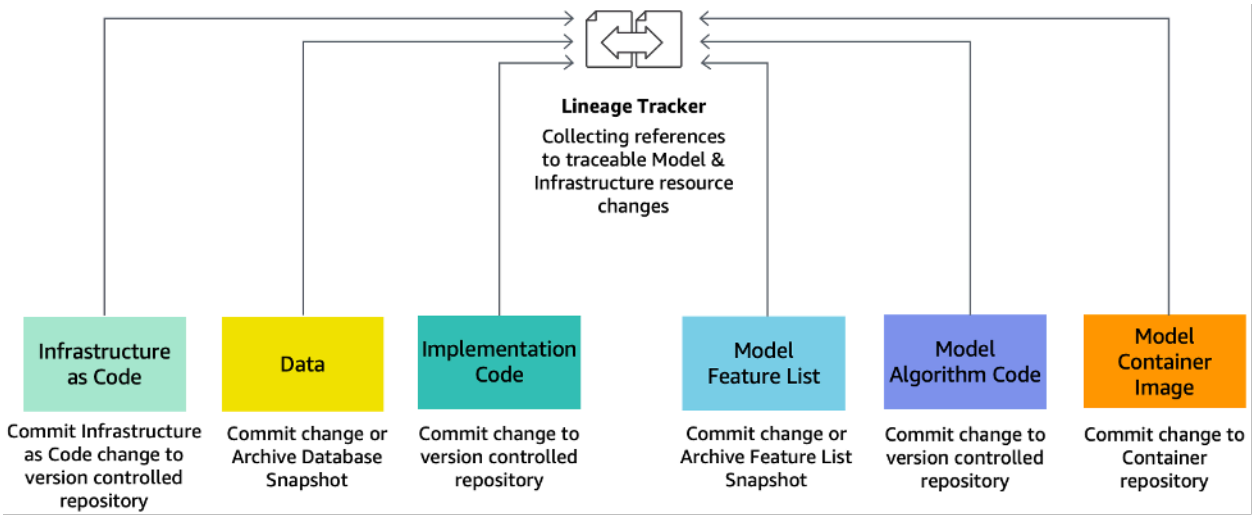

Figure 6: Lineage tracker

Lineage tracker components include:

- **Infrastructure as code (IaC)** - Modeling, provisioning, and managing cloud computing resources (compute, storage, network, and application services) can be automated using infrastructure as code. Cloud computing takes advantage of virtualization to enable the on-demand provisioning of resources. IaC eliminates configuration drift through automation, while increasing the speed and agility of infrastructure deployments. IaC code changes are committed to version-controlled repository.
- **Data** - Store data schemes and metadata in version control systems. Store the data in a storage media, such as a data lake. The location or link to the data can be in a configuration file and stored in code version control media.
- **Implementation code** - Changes to any implementation code at any point-in-time can be stored in version control media.
- **Model feature list** - A “feature store”, discussed earlier in this section (Figure 5), maintains the details of the features as well as their previous versions for any point-in-time changes.
- **Model algorithm code** - Changes to any model algorithm code at any point-in-time can be stored in version control media.
- **Model container image** - Versions of model container images for any point-in-time changes can be stored in container repositories managed by container registry.

ML lifecycle phase - Data processing

In ML workloads, the data (inputs and corresponding desired output) serves important functions including:

- Defining the goal of the system: the output representation and the relationship of each output to each input, by means of the input and output pairs.
- Training the algorithm that associates inputs to outputs.
- Measuring the performance of the model against changes in data distribution or data drift.
- Building a baseline dataset to capture data drift.

As shown in Figure 7, data processing consists of data collection and data preparation. Data preparation includes data preprocessing and feature engineering. It mainly uses data wrangling for interactive data analysis and data visualization for exploratory data analysis (EDA). EDA focuses on understanding data, sanity checks, and validation of data quality.

It is important to note that the same sequence of data processing steps that is applied to the training data needs to also be applied to the inference requests.

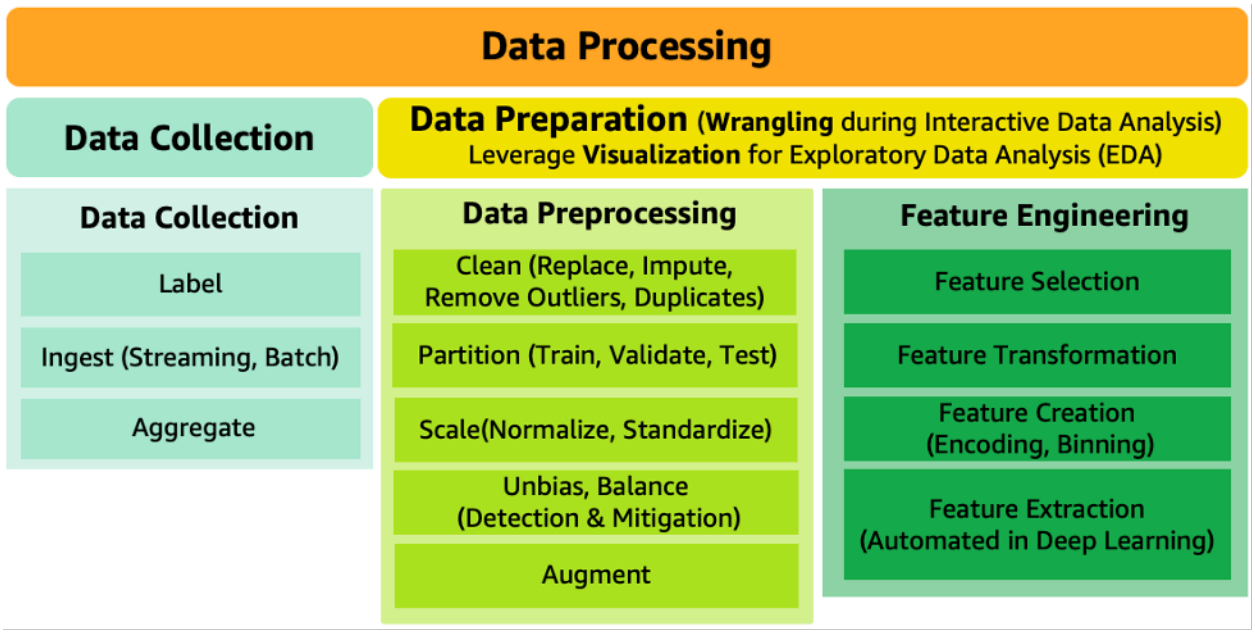

Figure 7: Data processing components

Best practices

- Data collection

- Data preparation
- Operational excellence pillar - Best practices
- Security pillar - Best practices
- Reliability pillar - Best practices
- Performance efficiency pillar - Best practices
- Cost optimization pillar - Best practices
- Sustainability pillar - Best practices

Data collection

Important steps in the ML lifecycle are to identify the data needed, followed by the evaluation of the various means available for collecting that data to train your model.

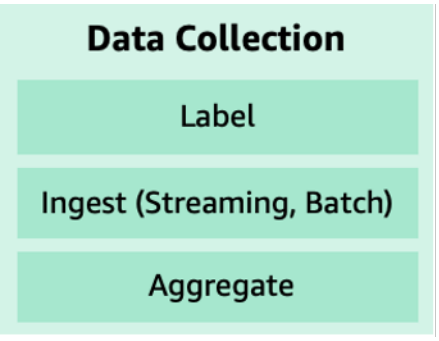

Figure 8: The main components of data collection

The main components of the data collection phase (shown in Figure 8) are as follows:

- **Label** - *Labeled data* is a group of samples that have been tagged with one or more labels. If labels are missing, then some effort is required to label it (either manual or automated).
- **Ingest and aggregate** - Data collection includes ingesting and aggregating data from multiple data sources.

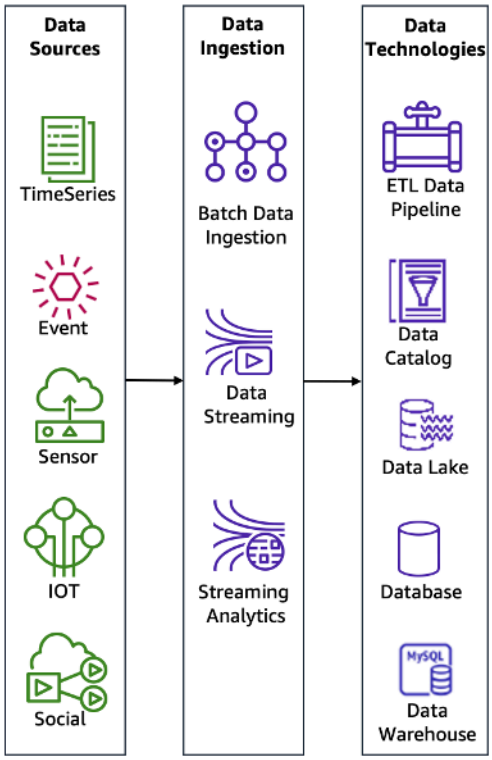

Figure 9: Data sources, data ingestion, and data technologies

The sub-components of the *ingest and aggregate* component (shown in Figure 9) are as follows:

- **Data sources** - Data sources include time-series, events, sensors, IoT devices, and social networks, depending on the nature of the use case. You can enrich your data sources by using the geospatial capability of Amazon SageMaker to access a range of geospatial data sources from AWS (for example, Amazon Location Service), open-source datasets (for example, Open Data on AWS), or your own proprietary data including from third-party providers (such as Planet Labs). To learn more about the geospatial capability in Amazon SageMaker, visit Geospatial ML with Amazon SageMaker (Preview).
- **Data ingestion**- Data ingestion processes and technologies capture and store data on storage media. Data ingestion can occur in real-time using streaming technologies or historical mode using batch technologies.
- **Data technologies** - Data storage technologies vary from transactional (SQL) databases, to data lakes and data warehouses. Extract, transform, and load (ETL) pipeline technology automates and orchestrates the data movement and transformations across cloud services and resources. A data lake technology enables storing and analyzing structured and unstructured data.

Data preparation

ML models are only as good as the data that is used to train them. Ensure that suitable training data is available and is optimized for learning and generalization. Data preparation includes data preprocessing and feature engineering.

A key aspect to understanding data is to identify patterns. These patterns are often not evident with data in tables. Exploratory data analysis (EDA) with visualization tools can help in quickly gaining a deeper understanding of data. Prepare data using data wrangler tools for interactive data analysis and model building. Employ no-code/low-code, automation, and visual capabilities to improve the productivity and reduce the cost for interactive analysis.

Data preprocessing

Data preprocessing puts data into the right shape and quality for training. There are many data preprocessing strategies including: data cleaning, balancing, replacing, imputing, partitioning, scaling, augmenting, and unbiasing.

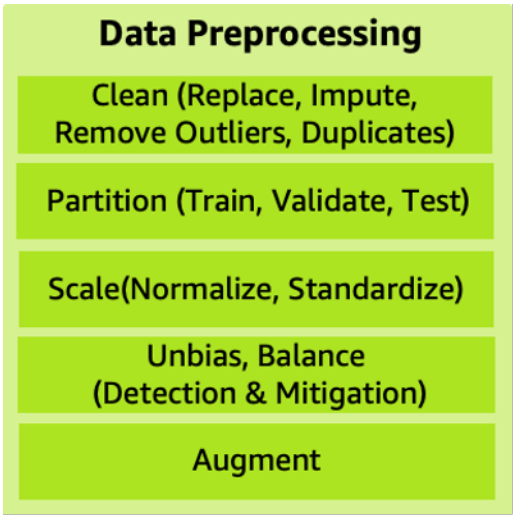

Figure 10: Data processing main components

The data preprocessing strategies listed in Figure 10 can be expanded as the following:

- **Clean (replace, impute, remove outliers and duplicates)** - Remove outliers and duplicates, replace inaccurate or irrelevant data, and correct missing data using imputation techniques that will minimize bias as part of data cleaning.
- **Partition** - To prevent ML models from overfitting and to evaluate a trained model accurately, randomly split data into train, validate, and test sets. Data leakage can happen when information

from hold-out test dataset leaks into the training data. One way to avoid data leakage is to remove duplicates before splitting the data.

- **Scale (normalize, standardize)** - Normalization is a scaling technique in machine learning that is applied during data preparation to change the values of numeric columns in the dataset to use a common scale. This technique helps ensure that each feature of the machine learning model has equal feature importance when they have different ranges. Normalized numeric features will have values in the range of [0,1]. Standardized numeric features will have a mean of 0 and standard deviation of 1. Standardization helps in handling outliers.
- **Unbias, balance (detection & mitigation)** - Detecting and mitigating bias helps avoid inaccurate model results. Biases are imbalances in the accuracy of predictions across different groups, such as age or income bracket. Biases can come from the data or algorithm used to train your model.
- **Augment** - Data augmentation increases the amount of data artificially by synthesizing new data from existing data. Data augmentation can help regularize and reduce overfitting.

Feature engineering

Every unique attribute of the data is considered a “feature” (also known as “attribute”). For example, when designing a solution for predicting customer churn, the data used typically includes features such as customer location, age, income level, and recent purchases.

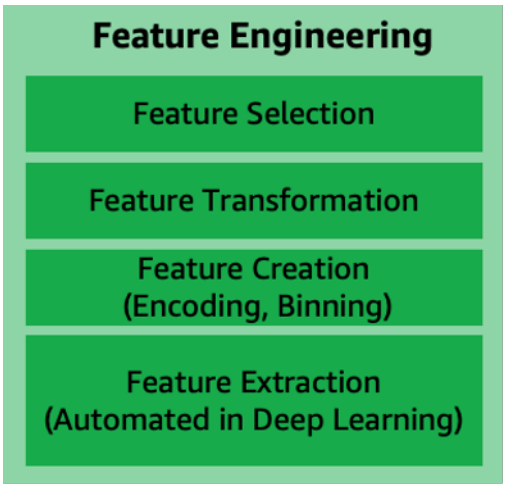

Figure 11: Feature engineering main components

Feature engineering is a process to select and transform variables when creating a predictive model using machine learning or statistical modeling. Feature engineering typically includes feature creation, feature transformation, feature extraction, and feature selection as listed in Figure 11. With deep learning, feature engineering is automated as part of the algorithm learning.

- **Feature creation** refers to the creation of new features from existing data to help with better predictions. Examples of feature creation include: one-hot-encoding, binning, splitting, and calculated features.
- **Feature transformation and imputation** include steps for replacing missing features or features that are not valid. Some techniques include: forming Cartesian products of features, non-linear transformations (such as binning numeric variables into categories), and creating domain-specific features.
- **Feature extraction** involves reducing the amount of data to be processed using dimensionality reduction techniques. These techniques include: Principal Components Analysis (PCA), Independent Component Analysis (ICA), and Linear Discriminant Analysis (LDA). This reduces the amount of memory and computing power required, while still accurately maintaining original data characteristics.
- **Feature selection** is the process of selecting a subset of extracted features. This is the subset that is relevant and contributes to minimizing the error rate of a trained model. Feature importance score and correlation matrix can be factors in selecting the most relevant features for model training.

Operational excellence pillar - Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and continually improve supporting processes. This section includes best practices to consider while processing data.

Best practices

- MLOE-10: Profile data to improve quality
- MLOE-11: Create tracking and version control mechanisms

MLOE-10: Profile data to improve quality

Profile data to use data characteristics like distribution, descriptive statistics, data types, and data patterns. Review source data for content and quality. Filter out or correct any data not passing the reviews. This will contribute to quality improvement.

Implementation plan

- Use the built-in data preparation capability of Amazon SageMaker Studio Notebook - This allows you to visually review data characteristics and remediate data-quality problems directly in your notebook environment. When you display a data frame (that is, a tabular representation of data) in your notebook, Amazon SageMaker Studio Notebook automatically generates charts to help users identify data-quality issues and suggests data transformations to help fix common problems. After you select a data transformation, Amazon SageMaker Studio Notebook generates the corresponding code within the notebook so that it can be repeatedly applied every time the notebook is run.
- **Use Amazon SageMaker Data Wrangler** - Import, prepare, transform, visualize, and analyze data with SageMaker Data Wrangler. You can integrate Data Wrangler into your ML workflows to simplify and streamline data pre-processing and feature engineering with little to no coding. You can also add your own Python scripts and transformations to customize your data preparation workflow. Import data from Amazon S3, Amazon Redshift, or other data sources, and then query the data using Amazon Athena. Use Data Wrangler to create sophisticated machine learning data preparation workflows with built-in and custom data transformations and analysis features. These features include feature target leakage and quick modeling.
- **Create an automatic data profile and a reporting system** - Use AWS Glue Crawler to crawl the data sources and create a data schema. Use AWSAWS Glue Data Catalog to list all the tables and schemas. Use Amazon Athena for serverless SQL querying to constantly profile data and then use Amazon QuickSight dashboards for visualization of the data.
- **Create a baseline dataset with SageMaker Model Monitor** – The training dataset used to train the model is usually a good baseline dataset. The training dataset data schema and the inference dataset schema should exactly match (the number and order of the features).

Documents

- Amazon SageMaker Notebooks
- Data Wrangler – Getting Started
- SageMaker Model Monitor – Create baseline

Blogs

- Next Generation SageMaker Notebooks – Now with Built-in Data Preparation, Real-Time Collaboration, and Notebook Automation

- Introducing Amazon SageMaker Data Wrangler, a Visual Interface to Prepare Data for Machine Learning
- Exploratory data analysis, feature engineering, and operationalizing your data flow into your ML with Amazon SageMaker Data Wrangler
- Prepare data for predicting credit risk using Amazon SageMaker Data Wrangler and Amazon SageMaker Clarify
- Prepare data from Snowflake for machine learning with Amazon SageMaker Data Wrangler
- Develop and deploy ML models using Amazon SageMaker Data Wrangler and Amazon SageMaker Autopilot
- Build an automatic data profiling and reporting solution with Amazon EMR, AWS Glue, and Amazon QuickSight
- Prepare image data with Amazon SageMaker Data Wrangler

MLOE-11: Create tracking and version control mechanisms

Due to its exploratory and iterative nature, it's easy to lose track of ML model development and its evolution. You need to experiment with multiple combinations of data, algorithms, and parameters, all while observing the impact of incremental changes on model accuracy. Log and track your model experiments with configuration settings and hyperparameters. Document and version control any data processing-related findings, processes, and improvement to enable easier future referencing and reuse. Use a model registry to register and version control your ML models. Automate your model deployment with CI/CD processes. To learn more about knowledge management, refer the best practice documented in OPS11-BP04.

Implementation plan

- **Track your ML experiments with SageMaker Experiments** - Amazon SageMaker Experiments lets you create, manage, analyze, and compare your machine learning experiments. SageMaker Experiments automatically tracks the inputs, parameters, configurations, and results of your iterations as runs. You can assign, group, and organize these runs into experiments. SageMaker Experiments is integrated with Amazon SageMaker Studio, providing a visual interface to browse your active and past experiments, compare runs on key performance metrics, and identify the best performing models
- **Associate notebook instances with Git repositories** - To analyze data, document its processing, and evaluate ML models on Amazon SageMaker, you can use Amazon SageMaker Processing. SageMaker Processing can be used for feature engineering, data validation, model evaluation,

and model interpretation. SageMaker notebook instances can be associated with Git repositories. This enables saving notebooks in a source control environment that persists after stopping or deleting the notebook instance. The notebooks hold the data processing code and its documentation. You can associate a default repository and up to three additional repositories with a notebook instance. The repositories can be hosted in AWS CodeCommit, GitHub, or on any other Git server.

- **Use SageMaker Model Registry** - Catalog, manage, and deploy models using SageMaker Model Registry. Create a model group and, for each run of your ML pipeline, create a model version that you register in the model group.

Documents

- Amazon SageMaker – Process Data
- Associate Git Repositories with SageMaker Notebook Instances

Blogs

- Amazon SageMaker notebooks now support Git integration for increased persistence, collaboration, and reproducibility
- How to manage Amazon SageMaker code with AWS CodeCommit
- Track your ML experiments end to end with Data Version Control and Amazon SageMaker Experiments

Examples

- Amazon SageMaker processing

Security pillar - Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve security.

Best practices

- MLSEC-03: Ensure least privilege access
- MLSEC-04: Secure data and modeling environment

- MLSEC-05: Protect sensitive data privacy
- MLSEC-06: Enforce data lineage
- MLSEC-07: Keep only relevant data

MLSEC-03: Ensure least privilege access

Protect all resources across various phases of the ML lifecycle using the principle of least privilege. These resources include: data, algorithms, code, hyperparameters, trained model artifacts, and infrastructure. Provide dedicated network environments with dedicated resources and services to operate any individual project.

Implementation plan

- **Restrict access based on business roles for individuals** - Identify roles that need to explore data to build models, features, and algorithms. Map those roles to access patterns using role-based authentication. This approach helps you achieve least privilege access to sensitive data, assets, and services on a project-by-project basis.
- **Use account separation and AWS Organizations** - Establish tagging and role-based access grants. Understand workflows of the different user types. Use Service Catalog to create pre-provisioned environments for quick deployment including a multi-account architecture that segregates workloads between development, test, and production with appropriate governance based on data sensitivity and compliance requirements. Tag data and buckets that contain sensitive workloads. Use these tags to grant granular access to individuals.
- **Break out ML workloads by access pattern and structure organizational units** - Delegate specific access to each group, such as administrators or data analysts, as required. Use guardrails and service control policies (SCPs) to enforce best practices for each access type and group. Limit infrastructure access to administrators. Verify all sensitive data is accessed through restricted, dedicated, and isolated environments.

Documents

- Amazon SageMaker with Guardrails on AWS
- Service Catalog
- Build a Secure Enterprise Machine Learning Platform on AWS
- AWS Well Architected Framework Security Pillar : Protecting Data at Rest

Blogs

- Building secure Amazon SageMaker access URLs with Service Catalog
- Setting up secure, well-governed machine learning environments on AWS - for detailed guidance on SCP and OU strategies

Videos

- AWS re:Invent 2020: Architectural best practices for machine learning applications
- AWS re:Invent 2020: Secure and compliant machine learning for regulated industries
- AWSre:Inforce 2019: Amazon SageMaker Model Development in a Highly Regulated Environment (SDD315)

Examples

- Build your own Anomaly Detection ML Pipeline
- AWS MLOps Framework

MLSEC-04: Secure data and modeling environment

Secure any system or environment that hosts data or enables model development. Store training data in secured storage and repositories. Run data preparation in a secure cloud. Tightly control access to the destination compute instances as data moves from the data repositories to the instances. Encrypt data at rest in the storage infrastructure and in transit to the compute infrastructure.

Implementation plan

- **Build a secure analysis environment** - During the data preparation and feature engineering phases, there are multiple options for secure data exploration on AWS. Data can be explored in an Amazon SageMaker managed notebook environment, or in an Amazon EMR notebook. You can also use managed services, such as Amazon Athena and AWS Glue, or a combination of the two, to explore the data without moving the data out of your data lake. Use an Amazon SageMaker Jupyter notebook instance to explore, visualize, and feature engineer a small subset of data. Scale up the feature engineering using a managed ETL service, such as Amazon EMR or AWS Glue.

- **Create dedicated AWS IAM and AWS KMS resources** – This approach limits the scope of impact of credentials and keys. Create a private S3 bucket and enable version control for the data and intellectual property (IP). In AWS, a centralized data lake is implemented using AWS Lake Formation on Amazon S3. Securing and monitoring a data lake on Amazon S3 is achieved using a combination of services and capabilities to encrypt data in transit and at rest. Monitor access using granular AWS IAM policies, S3 bucket policies, S3 Access Logs, Amazon CloudWatch, and AWS CloudTrail.
- **Use Secrets Manager and Paramater Store to protect credentials** - Secrets Manager enables you to replace hard-coded secrets in your code, such as credentials, with an API call to decrypt and retrieve the secret programmatically. Parameter Store was designed for wider use cases than secrets or passwords, but allows you to store application configuration variables such as AMI IDs or license keys. With AWS Secrets Manager and Parameter Store, you can store your credentials, and then grant permissions to your SageMaker IAM role to access Secrets Manager from your notebook.
- **Automate managing configuration** - Use lifecycle configurations scripts to manage Jupyter notebook instances. The scripts run when the notebook instance is first created, or every time it starts. They enable you to install custom packages, preload datasets, and set up source code repositories. Lifecycle configurations can be changed and reused across multiple notebook instances. You can make a change once and apply the updated configuration by restarting the managed notebook instances. This gives IT, operations, and security teams the flexibility and control they need, while supporting the needs of your developers and data scientists. Use AWS CloudFormation infrastructure as code, as well as Service Catalog to simplify configuration for end users.
- **Create private, isolated, network environments** - Use Amazon Virtual Private Cloud (Amazon VPC) to enable connectivity to only the services and users you need. Deploy the Amazon SageMaker notebook instance in an Amazon VPC to enable network level controls to limit communication to the hosted notebook. Additionally, network calls into and out of the notebook instance can be captured in VPC Flow Logs to enable additional visibility and control at the network level. By deploying the notebook in your VPC, you will also be able to query data sources and systems accessible from within your VPC, such as relational databases in Amazon RDS or Amazon Redshift data warehouses. Using IAM, you can further restrict access to the web-based UI of the notebook instance so that it can only be accessed from within your VPC. Use AWS PrivateLink to privately connect your SageMaker notebook instance VPC with supported AWS services. This ensures secure communication between your notebook instance and Amazon S3 within the AWS network. Use AWS KMS to encrypt data on the EBS volumes attached to SageMaker notebook instances.

- **Restrict access** - The Jupyter notebook server provides web-based access to the underlying operating system on an EC2 instance. This gives you the ability to install additional software packages or Jupyter kernels to customize your environment. The access is granted by default to a user with root access or super user on the operating system, giving them total control of the underlying EC2 instance. This access should be restricted to remove the user's ability to assume root permissions but still give them control over their local user's environment.
- **Secure ML algorithms** - Amazon SageMaker uses container technology to train and host algorithms and models. When creating your own containers, publish them to a private container registry hosted on Amazon Elastic Container Repository (Amazon ECR). Encrypt containers that are hosted on Amazon ECR at rest using AWS KMS.
- **Enforce code best practices** - Use secure git repositories through fully managed AWS CodeCommit for storing code.
- **Implement a package mirror for consuming approved packages** - Evaluate the license terms to determine which ML packages are appropriate for your business across the phases of the ML lifecycle. Examples of ML Python packages include: Pandas, PyTorch, Keras, NumPy, and Scikit-learn. Once you've determined the set and criteria, build a validation mechanism and automate it where possible. A sample automated mechanism can include a script that runs the download, installation, and package version and dependency checks. Only download packages from approved and private repos. Validate what is in the packages downloaded. This will enable importing safely and confirming the validity of packages. Amazon SageMaker notebook instances come with multiple environments already installed. These environments contain Jupyter kernels and Python packages. You can also install your own environments that contain your choice of packages and kernels. SageMaker enables modifying package channel paths to a private repository. Where appropriate, use an internal repository as a proxy for public repositories to minimize the network and time overhead.

Documents

- Amazon Sagemaker Workshop - Using Secure Environments
- Storage Best Practices for Data and Analytics Applications
- What is AWS CodeCommit?
- Security in Amazon SageMaker
- Amazon Well-Architected Security Pillar for Software Integrity
- Installing External Libraries and Kernels on Notebook Instances
- AWS Well Architected Framework Security Pillar : Protecting Data in Transit

Blogs

- 7 ways to improve security of your machine learning
- Building secure machine learning environments with Amazon SageMaker
- Setting up secure, well-governed machine learning environments on AWS
- Private package installation in Amazon SageMaker running internet-free mode
- Secure Deployment of Amazon SageMaker resource
- Create a hosting VPC for PyPi package mirroring and consumption of approved packages
- Apply fine-grained data access controls with AWS Lake Formation and Amazon EMR from Amazon SageMaker Studio

Videos

- Security for AI/ML Models in AWS
- AWS re:Invent 2020: Security best practices the AWS Well-Architected way

Examples

- Secure Data Science Reference Architecture

MLSEC-05: Protect sensitive data privacy

Protect sensitive data used in training against unintended disclosure. Identify and classify the sensitive data. Handle the sensitive data using strategies including: removing, masking, tokenizing, and principal component analysis (PCA). Document best governance practices for future reuse and references.

Implementation plan

- **Use automated mechanisms to classify data where possible** - Use automated sensitive data discovery in Amazon Macie that provides continual, cost efficient, organization-wide visibility into where sensitive data resides across your Amazon S3 environment. Macie automatically and intelligently inspects your S3 buckets for sensitive data such as personally identifiable information (PII), financial data, and AWS credentials. Macie then builds and continuously maintains an interactive data map of the locations in Amazon S3 where your sensitive data resides, and provides a sensitivity score for each bucket.

- **Use tagging** – Tag resources and models that are made from sensitive elements to quickly differentiate between resources requiring protection and those that do not.
- **Encrypt sensitive data** - Encrypt sensitive data using services such as AWS KMS, the AWS Encryption SDK, or client-side encryption.
- **Reduce data sensitivity** - Evaluate and identify data for anonymization or de-identification to reduce sensitivity.

Documents

- Running sensitive data discovery jobs in Amazon Macie
- Categorizing your storage using tags
- AWS Key Management Service best practices
- Getting started with the AWS Encryption SDK

Blogs

- 7 ways to improve security of your machine learning workflows
- Macie for Data Classification
- Building a Serverless Tokenization Solution to Mask Sensitive Data

Examples

- Amazon SageMaker Solution for Privacy in Natural Language Processing
- How Amazon is advancing privacy-aware data processing

MLSEC-06: Enforce data lineage

Monitor and track data origins and transformations over time. Strictly control data access. Perform preventative controls, auditing, and monitoring to demonstrate data lineage. Implement integrity checks against training data to detect any unexpected deviances caused by loss, corruption, or manipulation. Data lineage enables visibility and helps tracing root cause of data processing errors.

Implementation plan

- **Track records for any update** - Create and store information about the steps of a ML workflow from data preparation to model deployment using Amazon SageMaker ML Lineage Tracker. With the tracking information you can:
  - Reproduce the workflow steps, track model and dataset lineage, and establish model governance and audit standards.
  - Consider origin data to be the source of truth.
  - Ingest and process derived datasets and retain mappings throughout the process. Iterate from the end result back to the original data element.
  - Apply these concepts not just to data, but also the code, models, pipelines, and infrastructure. Validate that you can trace and audit any activity against data, pipeline actions, or machine learning models.

Documents

- Amazon SageMaker ML Lineage Tracking

Blogs

- Using model attributes to track your training runs on Amazon SageMaker

Examples

- LAB 04: DevOps WorkFlow

MLSEC-07: Keep only relevant data

Preserve data across computing environments (such as development and staging) and only store use-case relevant data to reduce data exposure risks. Implement mechanisms to enforce a lifecycle management process across the data. Decide when to automatically remove stale data.

Implementation plan

- **Establish a data lifecycle plan** - Understand usage patterns and requirements for debugging and operational tasks. Establish a data lifecycle plan to reduce data sprawl over time.

- **Design for privacy** - Remove sensitive elements that are not needed for the ML workflow. Detect and redact personally identifiable information (PII), while maintaining data usability. Determine what features are required to solve the business problem and valuable for future iterations.

Documents

- Reference Guide: Extract More Value from your Data

Blogs

- Building a data analytics practice across the data lifecycle
- Detecting and redacting PII using Amazon Comprehend
- Now available in Amazon Transcribe: Automatic Redaction of Personally Identifiable Information
- Machine learning models that act on encrypted data
- Redacting sensitive information with user-defined functions in Amazon Athena

Videos

- AWS re:Invent 2020: Privacy-preserving machine learning
- AWS re:Invent 2019: Best practices for Amazon S3

Examples

- Field Notes: Redacting Personal Data from Connected Cars Using Amazon Rekognition
- How to Create a Modern CPG Data Architecture with Data Mesh

Reliability pillar - Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This section includes best practices to consider while processing data.

Best practices

- MLREL-03: Use a data catalog
- MLREL-04: Use a data pipeline

- MLREL-05: Automate managing data changes

MLREL-03: Use a data catalog

Process data across multiple data stores using data catalog technology. An advanced data catalog service can enable ETL process integration. This approach enables more reliability and efficiency.

Implementation plan

- **Use AWS Glue Data Catalog** - The AWS Glue Data Catalog provides a way to track the data assets that have been loaded into your ML workload. Data catalogs also describe how data is transformed as it is loaded into the data lake and data warehouse. AWS Glue is a fully managed ETL (extract, transform, and load) service. It enables a simple and cost-effective approach to categorize your data, clean it, enrich it, and move it reliably between various data stores and data streams. AWS Glue consists of a central metadata repository known as the AWS Glue Data Catalog. It also has an ETL engine that automatically generates Python or Scala code. With a flexible scheduler, AWS Glue handles dependency resolution, job monitoring, and retries.

Documents

- Data Cataloging
- Populating the AWSAWS Glue Data Catalog

Blogs

- Moving from notebooks to automated ML pipelines using Amazon SageMaker and AWS Glue
- How Genworth built a serverless ML pipeline on AWS using Amazon SageMaker and AWS Glue

Videos

- Getting Started with AWS Glue Data Catalog
- AWS re:Invent 2018: How Bill.com Uses Amazon SageMaker & AWS Glue to Enable Machine Learning - STP10
- AWS re:Invent 2018: Build and Govern Your Data Lakes with AWS Glue (ANT309)

Examples

- Explaining Credit Decisions with Amazon SageMaker
- How to build an end-to-end Machine Learning pipeline using AWS Glue, Amazon S3, Amazon SageMaker and Amazon Athena.

MLREL-04: Use a data pipeline

Automate the processing, movement, and transformation of data between different compute and storage services. This automation enables data processing that is fault tolerant, repeatable, and highly available.

Implementation plan

- **Use Amazon SageMaker Data Wrangler and Pipelines** - SageMaker Data Wrangler simplifies the preparation of machine learning data. It enables data selection, cleansing, exploration, and visualization using a single visual interface. After you've created a workflow, export it to SageMaker Pipelines to automate model deployment and management. Data pipelines provide an automated way to move and transform data in your ML workload. Manually moving and transforming data can lead to errors and inconsistencies. Use AWS Data Pipeline to save time and decrease errors.

Documents

- Prepare ML Data with Amazon SageMaker Data Wrangler
- Amazon SageMaker Model Building Pipelines

Blogs

- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
- Develop and deploy ML models using Amazon SageMaker Data Wrangler and Amazon SageMaker Autopilot

Videos

- AWS re:Invent 2020: Accelerate data preparation with Amazon SageMaker Data Wrangler
- Amazon SageMaker Data Wrangler Deep Dive Demo

Examples

- Amazon SageMaker Data Wrangler and Feature Store
- SageMaker Pipelines

MLREL-05: Automate managing data changes

Automate managing changes to training data using version control technology. This will enable reproducibility to re-create the exact version of a model in the event of a failure.

Implementation plan

- **Use AWS MLOps Framework** - AWS MLOps Framework provides a standard interface for managing ML pipelines for Amazon Machine Learning services and third-party services. The solution's template allows you to upload your trained models (also referred to as *bring your own model*). It configures the orchestration of the pipeline, and monitors the pipeline's operations. This solution increases agility and efficiency by allowing repeating of successful processes at scale. One of the key components of MLOps pipeline in SageMaker is Model Registry. SageMaker Model Registry tracks the model versions and respective artifacts, including the lineage and metadata.

Documents

- AWS MLOps Framework
- Register and Deploy Models with Model Registry

Blogs

- Amazon SageMaker Pipelines Brings DevOps Capabilities to your Machine Learning Projects

Videos

- Solving with AWS Solutions: AWS MLOps Framework

Examples

- Amazon SageMaker secure MLOps

Performance efficiency pillar - Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve. This section includes best practices to consider while processing data.

Best practices

- MLPER-04: Use a modern data architecture

MLPER-04: Use a modern data architecture

Get the best insights from exponentially growing data using a modern data architecture. This architecture enables easy movement of data between a data lake and purpose-built stores including a data warehouse, relational databases, non-relational databases, ML and big data processing, and log analytics. A data lake provides a single place to run analytics across mixed data structures collected from disparate sources. Purpose-built analytics services provide the speed required for specific use cases like real-time dashboards and log analytics.

Implementation plan

- **Unify data governance and access** - Integrate a data lake, a data warehouse, and purpose-built stores. This will enable unified governance and easy data movement. With a Modern Data Architecture on AWS, you can store data in a data lake and use data services around it. Use AWS Lake Formation to build a scalable and secure data lake. Build a high-speed analytic layer with purpose-built services, such as Amazon Redshift, Amazon Kinesis, and Amazon Athena. Integrate data across services and data stores with AWS Glue. Apply governance policies to manage security, access control, and audit trails across all the data stores using AWS IAM.

Documents

- Data Lake on AWS
- AWS Lake Formation
- Derive Insights from Modern Data

Blogs

- Build a Lake House Architecture on AWS

- Moving from notebooks to automated ML pipelines using Amazon SageMaker and AWS Glue
- Data preprocessing for machine learning on Amazon EMR made easy with AWS Glue DataBrew

Videos

- Build and Govern Your Data Lakes with AWS Glue
- The lake house approach to data warehousing with Amazon Redshift

Examples

- Predictive Data Science with Amazon SageMaker and a Data Lake on AWS

Cost optimization pillar - Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment. This section includes best practices to consider while processing data.

Best practices

- MLCOST-05: Use managed data labeling
- MLCOST-06: Use data wrangler tools for interactive analysis
- MLCOST-07: Use managed data processing capabilities
- MLCOST-08: Enable feature reusability

MLCOST-05: Use managed data labeling

Choose a managed labeling tool that provides automation and access to cost-effective teams of human data labelers . It should also provide flexibility to choose a variable number of labelers for a given input. The tool should have a user interface, and learn to label data by itself over time.

Implementation plan

- **Use Amazon SageMaker Ground Truth** - To train a machine learning model, you need a large, high quality, labeled dataset. Amazon SageMaker Ground Truth helps you build high-quality training datasets for your ML models. With Ground Truth, you can use ML along with workers from Amazon Mechanical Turk, a vendor company that you choose, or an internal, private workforce to create a labeled dataset. You can use the labeled dataset output from Ground Truth to train your own models. You can also use the output as a training data set for an Amazon SageMaker model.
- **Use Amazon SageMaker Ground Truth Plus** – Ground Truth Plus is a turn-key service that uses an expert workforce to deliver high-quality training datasets fast, and reduces costs by up to 40 percent. Amazon SageMaker Ground Truth Plus enables you to easily create high-quality training datasets without having to build labeling applications and manage the labeling workforce on your own. By using this approach, you don't need to have deep ML expertise or extensive knowledge of workflow design and quality management. You simply provide data along with labeling requirements and Ground Truth Plus sets up the data labeling workflows and manages them on your behalf in accordance with your requirements.

Documents

- Use Amazon SageMaker Ground Truth to Label Data
- Use Amazon SageMaker Ground Truth Plus to Label Data

Blogs

- Real-time data labeling pipeline for ML workflows using Amazon SageMaker Ground Truth
- Implementing a custom labeling GUI with built-in processing logic with Amazon SageMaker Ground Truth
- Amazon SageMaker Ground Truth Plus – Create Training Datasets Without Code or In-house Resources
- Get Started with Amazon SageMaker Ground Truth Plus

Videos

- Amazon SageMaker Ground Truth Plus

Examples

- Bring your own model for SageMaker labeling workflows with active learning
- SageMaker Ground Truth recipe
- How to setup and use SageMaker Ground Truth

MLCOST-06: Use data wrangler tools for interactive analysis

Prepare data through data wrangler tools for interactive data analysis and model building. The no-code/low-code, automation, and visual capabilities improve the productivity and reduce the cost for interactive analysis.

Implementation plan

- **Use Amazon SageMaker Data Wrangler** - Amazon SageMaker Data Wrangler, part of SageMaker Studio, provides an end-to-end solution to import, prepare, transform, analyze data and select features. You can integrate a Data Wrangler data flow into your ML workflows to simplify and streamline data pre-processing and feature engineering using little to no coding. You can also add your own Python scripts and transformations to customize workflows.

Documents

- Amazon SageMaker Data Wrangler

Blogs

- Introducing Amazon SageMaker Data Wrangler, a Visual Interface to Prepare Data for Machine Learning
- Develop and deploy ML models using Amazon SageMaker Data Wrangler and Amazon SageMaker Autopilot
- Prepare time series data with Amazon SageMaker Data Wrangler
- Accelerate data preparation with data quality and insights in Amazon SageMaker Data Wrangler
- Balance your data for machine learning with Amazon SageMaker Data Wrangler
- Process larger and wider datasets with Amazon SageMaker Data Wrangler

Videos

- Amazon SageMaker Data Wrangler Deep Dive Demo

Examples

- Prepare ML Data with Amazon SageMaker Data Wrangler

MLCOST-07: Use managed data processing capabilities

With managed data processing, you can use a simplified, managed experience to run your data processing workloads, such as feature engineering, data validation, model evaluation, and model interpretation.

Implementation plan

- **Use Amazon SageMaker Processing –** With Amazon SageMaker Processing, you can run processing jobs for data processing steps in your machine learning pipeline. Processing jobs accept data from Amazon S3 as input and store data into Amazon S3 as output. The processing container image can either be an Amazon SageMaker built-in image or a custom image that you provide. The underlying infrastructure for a Processing job is fully managed by Amazon SageMaker. Cluster resources are provisioned for the duration of your job, and cleaned up when a job completes. SageMaker Processing has simplified running machine learning preprocessing and postprocessing tasks with popular frameworks such as scikit-learn, Apache Spark, PyTorch, TensorFlow, Hugging Face, MXNet, and XGBoost.

Documents

- Process Data with SageMaker Processing

Blogs

- Amazon SageMaker Processing – Fully Managed Data Processing and Model Evaluation
- Use deep learning frameworks natively in Amazon SageMaker Processing
- Building machine learning workflows with Amazon SageMaker Processing jobs and AWS Step Functions

- Process Amazon Redshift data and schedule a training pipeline with Amazon SageMaker Processing and Amazon SageMaker Pipelines

Examples

- Amazon SageMaker Processing jobs

MLCOST-08: Enable feature reusability

Reduce duplication and the rerunning of feature engineering code across teams and projects by using feature storage. The store should have online and offline storage, and data encryption capabilities. An online store with low-latency retrieval capabilities is ideal for real-time inference. An offline store maintains a history of feature values and is suited for training and batch scoring.

Implementation plan

- **Use Amazon SageMaker Feature Store** - Amazon SageMaker Feature Store is a fully managed, purpose-built repository to store, update, retrieve, and share ML features. Feature Store makes it easy for data scientists, machine learning engineers, and general practitioners to create, share, and manage features for ML development. The online store is used for low latency, real-time inference use cases. The offline store is used for training and batch inference. The Feature Store reduces the repetitive data processing and curation work required to convert raw data into features for training an ML algorithm.

You can use Feature Store in the following modes:

- **Online** - Features are read with low latency reads (milliseconds) and used for high throughput predictions.
- **Offline** - Large streams of data are fed to an offline store, which is used for training and batch inference. This mode requires a feature group to be stored in an offline store. The offline store uses your S3 bucket for storage and can also fetch data using Amazon Athena queries.
- **Online and offline** - This includes both online and offline modes.

Documents

- Create, Store, and Share Features with Amazon SageMaker Feature Store

Blogs

- Getting started with Amazon SageMaker Feature Store
- Store, Discover, and Share Machine Learning Features with Amazon SageMaker Feature Store
- Enable feature reuse across accounts and teams using Amazon SageMaker Feature Store
- Understanding the key capabilities of Amazon SageMaker Feature Store
- Using Amazon SageMaker Feature Store with streaming feature aggregation
- Extend model lineage to include ML features using Amazon SageMaker Feature Store

Videos

- Amazon SageMaker Feature Store Deep Dive Demo

Examples

- Using Amazon SageMaker Feature Store with streaming feature aggregation
- Amazon SageMaker Feature Store Notebook Examples

Sustainability pillar - Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. This section includes best practices to consider while processing data.

Best practices

- MLSUS-04: Minimize idle resources
- MLSUS-05: Implement data lifecycle policies aligned with your sustainability goals
- MLSUS-06: Adopt sustainable storage options

Related best practices

- **Enable data and compute proximity** (MLCOST-21) The two most important factors influencing the carbon footprint of your network usage when transmitting data are the size of the data and the distance traveled. Compress your data before moving it over the network. Minimize data

movement across networks when selecting a Region. Store your data close to your producers and train your models close to your data.

- **Enable feature reusability** (MLCOST-08)- Evaluate if you can avoid data processing by using existing publicly available datasets like AWS Data Exchange and Open Data on AWS (which includes the Amazon Sustainability Data Initiative). They offer a variety of data, including weather and climate datasets, satellite imagery, air quality data, and energy data. By using these curated datasets, you avoid duplicating the compute and storage resources needed to download the data from the providers, store it in the cloud, organize, and clean it. For internal data, you can also reduce the duplication of feature engineering code across teams and projects by using managed feature storage services, such as Amazon SageMaker Feature Store.

MLSUS-04: Minimize idle resources

Adopt a managed and serverless architecture for your data pipeline so that it only provisions resources when work needs to be done. By doing so, you are not maintaining compute infrastructure 24/7 and you minimize idle resources.

Implementation plan

- **Use managed services** - Managed services shift responsibility for maintaining high average utilization, and sustainability optimization of the deployed hardware to AWS. Use managed services to distribute the sustainability impact of the service across all tenants of the service, reducing your individual contribution.
- **Create a serverless, event-driven data pipeline** - Use AWS Glue and AWS Step Functions for data ingestion and preprocessing. Step Functions can orchestrate AWS Glue jobs to create event-based serverless ETL and ELT pipelines. Because AWS Glue and AWS Step Functions are serverless, compute resources are only used as needed and not in an idle state while waiting.

Documents

- Manage AWS Glue Jobs with Step Functions

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data
- Centralize feature engineering with AWS Step Functions and AWS Glue DataBrew

- Optimizing your AWS Infrastructure for Sustainability, Part I: Compute

Metrics

- If using Amazon Elastic Compute Cloud (Amazon EC2), measure and optimize the CPU Utilization of the compute instances involved in data preparation.
- If using Amazon Elastic Container Service (Amazon ECS), measure and optimize the CPU Utilization used in the cluster or service.
- If using Amazon Elastic Kubernetes Service (Amazon EKS), measure and optimize the CPU Utilization of your nodes and pods.
- If using Amazon EMR, optimize the Idle time of the cluster.

MLSUS-05: Implement data lifecycle policies aligned with your sustainability goals

Classify data to understand its significance to your workload and your business outcomes. Use this information to determine when you can move data to more energy-efficient storage or safely delete it.

Define data retention periods that support your sustainability goals while meeting, but not exceeding, your business requirements.

Implementation plan

- **Define lifecycle policies for all your data classification types** - Determine the requirements for the retention and deletion of your data.
- **Manage the lifecycle of all your data** - Automatically enforce deletion timelines to minimize the total storage requirements of your workload using Amazon S3 Lifecycle policies.
- **Automatically optimize storage sustainability based on access patterns** - Use Amazon S3 Intelligent-Tiering storage class to automatically move your data to the most sustainable access tier when access patterns change.

Documents

- Implement a data classification policy
- Use lifecycle policies to delete unnecessary data

- Managing your storage lifecycle on Amazon S3
- Amazon S3 Intelligent-Tiering

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data

Metrics

- Measure and optimize the total size of your S3 buckets and storage class distribution, using Amazon S3 Storage Lens

MLSUS-06: Adopt sustainable storage options

Reduce the volume of data to be stored and adopt sustainable storage options to limit the carbon impact of your workload. For artifacts like models and log files that must be kept for long-term compliance and audit requirements, use efficient compression algorithms and use energy efficient cold storage.

Implementation plan

- **Reduce redundancy of processed data** - If you can easily re-create an infrequently accessed dataset, use the Amazon S3 One Zone-IA class to minimize the total data stored.
- **Right size block storage for notebooks** - Don't over-provision block storage of your notebooks and use centralized object storage services like Amazon S3 for common datasets to avoid data duplication.
- **Use efficient file formats** - Use Parquet or ORC to train your models. Compared to CSV, they can help you reduce your storage by up to 87%.
- **Migrate to more efficient compression algorithms** - Evaluate different compression algorithms and select the most efficient for your data. For example, Zstandard produces 10–15% smaller files than Gzip at the same compression speed.

Documents

- Use technologies that support data access and storage patterns

Blogs

- Optimize AI/ML workloads for sustainability: Part 1, identify business goals, validate ML use, and process data
- Optimizing your AWS Infrastructure for Sustainability, Part II: Storage
- Compressing and archiving logs to the Amazon S3 Glacier storage classes

Metrics

- Measure and optimize the total size of your S3 buckets and storage class distribution, using Amazon S3 Storage Lens
- If using SageMaker Studio, monitor and optimize the size of the shared Amazon Elastic File System (Amazon EFS) volume for the team.

ML lifecycle phase - Model development

Model development consists of model building, training, tuning, and evaluation.

Best practices

- Model training and tuning
- Model evaluation
- Operational excellence pillar - Best practices
- Security pillar – Best practices
- Reliability pillar – Best practices
- Performance efficiency pillar – Best practices
- Cost optimization pillar – Best practices
- Sustainability pillar – Best practices

Model training and tuning

In this phase, you select a machine learning algorithm that is appropriate for your problem and then train the ML model. You provide the algorithm with the training data, set an objective metric for the ML model to optimize on, and set the hyperparameters to optimize the training process.

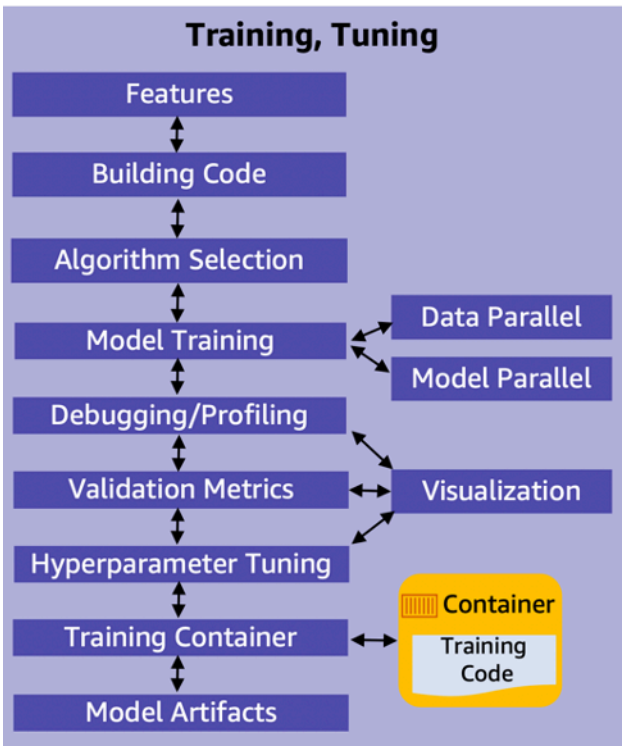

Figure 12: ML model training and tuning: main components

Model training, tuning, and evaluation require prepared data and engineered features. The following are the main activities in this stage, as listed in Figure 12:

- **Features** – Features are selected as part of the data processing after data quality is assured
- **Building code** – Model development includes building the algorithm and its supporting code. The code-building process should support version control and continuous build, test, and integration through a pipeline.
- **Algorithm selection** – Selecting the right algorithm involves running many experiments with parameter tuning across available options. Factors to consider when evaluating each option can include success metrics, model explainability, and compute requirements (training/prediction time and memory requirements).
- **Model training** (data parallel, model parallel)– The process of training a ML model involves providing the algorithm with training data to learn from. Distributed training enables splitting large models and training datasets across compute instances to reduce training time significantly. Model parallelism and data parallelism are techniques to achieve distributed training.
  - **Model parallelism** is the process of splitting a model up between multiple instances or nodes.

    - **Data parallelism** is the process of splitting the training set in mini-batches evenly distributed across nodes. Thus, each node only trains the model on a fraction of the total dataset.
- **Debugging/profiling** – A machine learning training job can have problems including: system bottlenecks, overfitting, saturated activation functions, and vanishing gradients. These problems can compromise model performance. A debugger provides visibility into the ML training process through monitoring, recording, and analyzing data. It captures the state of a training job at periodic intervals.
- **Validation metrics** – Typically, a training algorithm computes several metrics such as loss and prediction accuracy. These metrics determine if the model is learning and generalizing well for making predictions on unseen data. Metrics reported by the algorithm depend on the business problem and the ML technique used. For example, a *confusion matrix* is one of the metrics used for classification models, and Root Mean Squared Error (RMSE) is one of the metrics for regression models.
- **Hyperparameter tuning** – Settings that can be tuned to control the behavior of the ML algorithm are referred to as *hyperparameters*. The number and type of hyperparameters in ML algorithms are specific to each model. Examples of commonly used hyperparameters include: learning rate, number of epochs, hidden layers, hidden units and activation functions. Hyperparameter tuning, or optimization, is the process of choosing the optimal hyperparameters for an algorithm.
- **Training code container** – Create container images with your training code and its entire dependency stack. This will enable the training and deployment of models quickly and reliably at any scale.
- **Model artifacts**– Model artifacts are the outputs that results from training a model. They typically consist of trained parameters, a model definition that describes how to compute inferences, and other metadata.
- **Visualization** – Enables exploring and understanding data during metrics validation, debugging, profiling, and hyperparameter tuning.

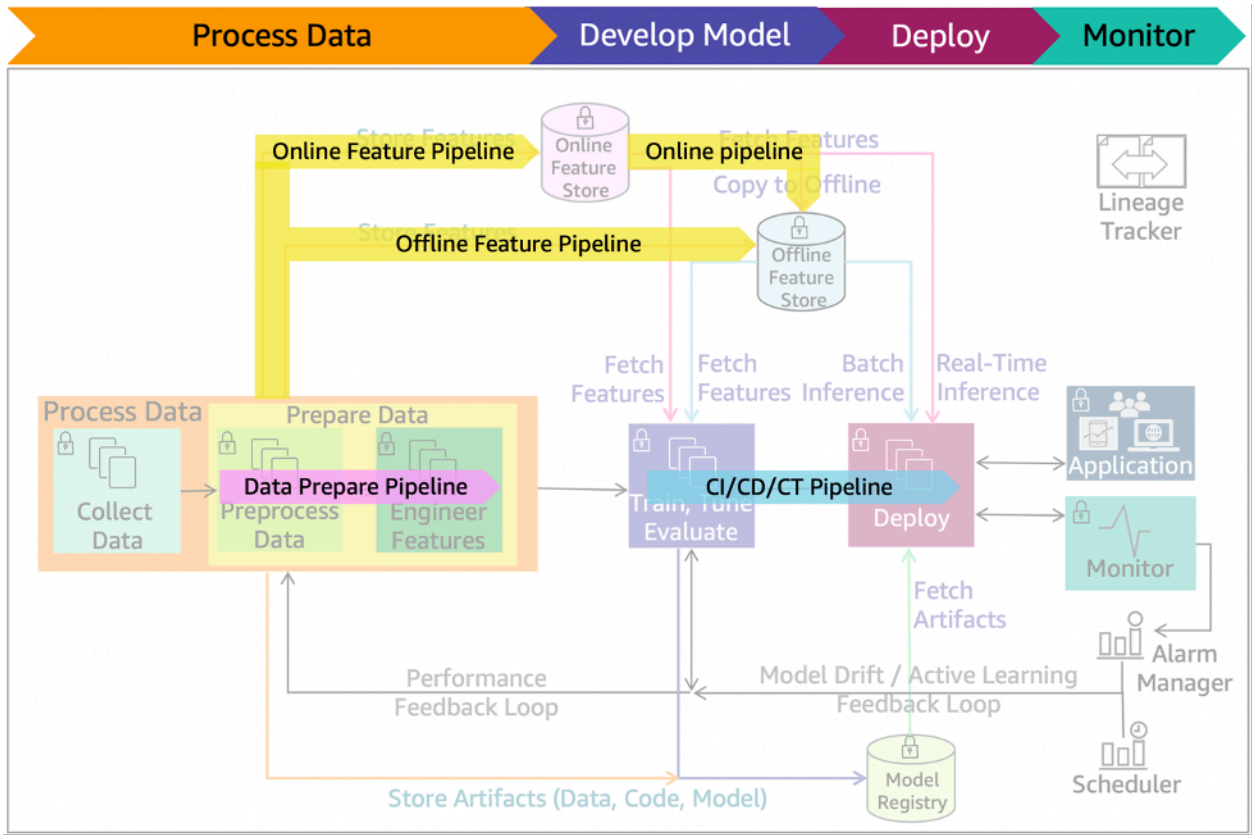

Figure 13: ML lifecycle with pre-production pipelines

Figure 13 shows the pre-production pipelines. The *data prepare* pipeline automates data preparation tasks. The feature pipeline automates the storing, fetching, and copying of the features into and from online/offline store. The CI/CD/CT pipeline automates the build, train, and release to staging and production environments.

Model evaluation

After the model has been trained, evaluate it for its performance and success metrics. You might want to generate multiple models using different methods and evaluate the effectiveness of each model. You also might evaluate whether your model must be more sensitive than specific, or more specific than sensitive. For multiclass models, determine error rates for each class separately.

You can evaluate your model using historical data (offline evaluation) or live data (online evaluation). In offline evaluation, the trained model is evaluated with a portion of the dataset that has been set aside as a *holdout set*. This holdout data is never used for model training or validation-it's only used to evaluate errors in the final model. The holdout data annotations must have high assigned label correctness for the evaluation to make sense. Allocate additional resources to verify the correctness of the holdout data.

Based on the evaluation results, you might fine-tune the data, the algorithm, or both. When you fine-tune the data, you apply the concepts of data cleansing, preparation and feature engineering.

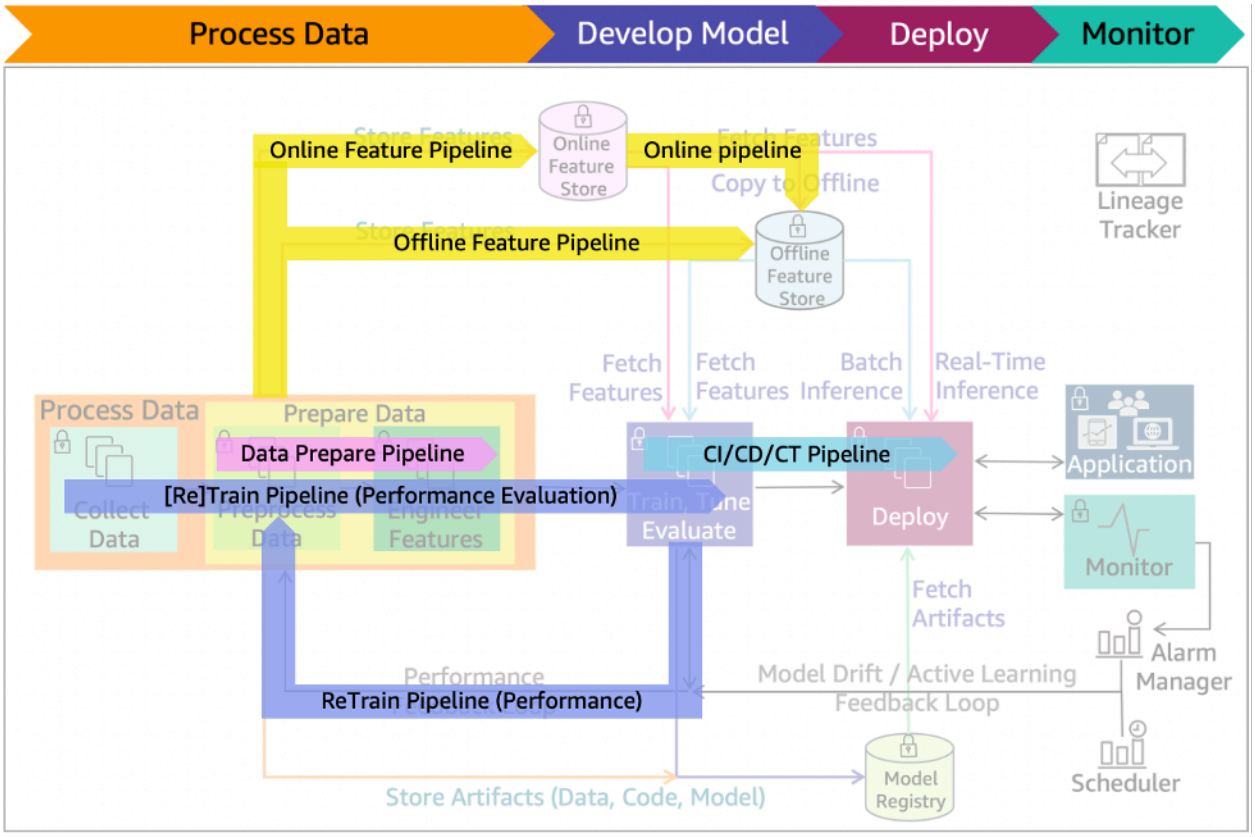

Figure 14: ML lifecycle with performance evaluation pipeline added

Figure 14 includes the model performance evaluation, the *data prepare* and CI/CD/CT pipelines that fine-tune data and/or algorithm, re-training, and evaluation of model results.

Operational excellence pillar - Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures. This section includes best practices to consider while developing models that includes training, tuning, and model performance evaluation.

Best practices

- MLOE-12: Automate operations through MLOps and CI/CD
- MLOE-13: Establish reliable packaging patterns to access approved public libraries

MLOE-12: Automate operations through MLOps and CI/CD

Automate ML workload operations using infrastructure as code (IaC) and configuration as code (CaC). Select appropriate MLOps mechanisms to orchestrate your ML workflows and integrate with CI/CD pipelines for automated deployments. This approach ensures consistency across your staging and production deployment environments. Enable model observability and version control across your hosting infrastructure.

Implementation plan

You can choose either AWS CloudFormation or AWS Cloud Development Kit (AWS CDK):

- **Use AWS CloudFormation** -AWS CloudFormation enables you to create and provision AWS deployments predictably and repeatedly by using a template file to create and delete a collection of resources together as a single unit (a stack). You can manage and provision stacks across multiple AWS accounts and AWS Regions.
- **Use AWS Cloud Development Kit (AWS CDK)** - Use AWS Cloud Development Kit (AWS CDK) (AWS CDK) as a software development framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation. You can define your cloud resources in AWS CDK using familiar programming languages.

You can choose any of the following MLOps strategies based on your ML workflows:

- **Use SageMaker Pipelines to orchestrate your workflows**

Using Amazon SageMaker Pipelines, you can create ML workflows with Python SDK, and then visualize and manage your workflow using Amazon SageMaker Studio. Amazon SageMaker Pipelines logs every step of your workflow, creating an audit trail of model components such as training data, platform configurations, model parameters, and learning gradients.

- **Use AWS Step Functions**- You can also use AWS Step Functions Data Science SDK for

Amazon SageMaker to automate training of a machine learning model. Define all the

steps in the workflow and set up alerts to start the flow.

- **Use third-party tools** - Use third-party deployment orchestration tools, such as

Apache Airflow, that integrate with AWS service APIs to automate model training and deployment. Amazon Managed Workflows for Apache Airflow (MWAA) orchestrates your workflows using Directed Acyclic Graphs (DAGs) written in Python.

data is available.

Documents

- Infrastructure as Code
- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK) (CDK)
- SageMaker Pipelines
- Step Functions Data Science SDK

Blogs

- AWS CloudFormation – Create Your AWS Stack from a Recipe
- Automate Amazon SageMaker Studio setup using AWS SDK
- Secure multi-account model deployment with Amazon SageMaker: Part 1
- Secure multi-account model deployment with Amazon SageMaker: Part 2
- Organize your machine learning journey with SageMaker Pipelines

MLOE-13: Establish reliable packaging patterns to access approved public libraries

Establish reliable packaging patterns for data scientists, which include (a) the use of internal repositories that provide access to public libraries and (b) the creation of separate kernels for common ML frameworks. Examples of such common ML frameworks include TensorFlow, PyTorch, Scikit-learn, and Keras.

Implementation plan

- **Use container technology** - Use or alternatively bring custom containers and store them in Amazon Elastic Container Registry (Amazon ECR). Using containers, you can train machine learning algorithms and deploy models quickly and reliably at any scale.
- **Use artifact repository** - Set up AWS CodeArtifact to be used as a central internal artifact repository. This will enable pulling artifacts from internal repositories and reusing them.

Documents

- Train a Deep Learning model with AWS Deep Learning Containers on Amazon EC2

Blogs

- Private package installation in Amazon SageMaker running in internet-free mode
- Bringing your own custom container image to Amazon SageMaker Studio notebooks
- Integrating Jenkins with AWS CodeArtifact to publish and consume Python artifacts

Security pillar – Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. This section includes best practices to consider during model development that includes model training, tuning, and model performance evaluation.

Best practices

- MLSEC-08: Secure governed ML environment
- MLSEC-09: Secure inter-node cluster communications
- MLSEC-10: Protect against data poisoning threats

MLSEC-08: Secure governed ML environment

Protect ML operations environments using managed services with best practices including: detective and preventive guardrails, monitoring, security, and incident management. Explore data in a managed and secure development environment. Centrally manage the configuration of development environments and enable self-service provisioning for the users.

Implementation plan

- **Break out ML workloads** by organizational unit access patterns. This will enable delegating required access to each group, such as administrators or data analysts.
- **Use guardrails and service control policies (SCPs)** to enforce best practices for each environment type. Limit infrastructure management access to administrators.

- **Verify all sensitive data has access through restricted, isolated environments**. Ensure network isolation, dedicated resources, and check service dependencies.
- **Secure ML algorithm implementation** using a restricted development environment. Secure model training and hosting containers by following the security processes required for your organization.

Documents

- Security in Amazon SageMaker
- Build a secure enterprise machine learning platform on AWS
- Use Amazon SageMaker Notebook Instances
- Amazon SageMaker with Guardrails on AWS

Blogs

- Setting up secure, well-governed machine learning environments on AWS
- Securing Amazon SageMaker Studio Connectivity using a private VPC
- Enable self-service, secured data science using Amazon SageMaker notebooks and AWS Service Catalog
- Accelerating Machine Learning Development with Data Science as a Service from Change Healthcare

MLSEC-09: Secure inter-node cluster communications

For frameworks such as TensorFlow, it's common to share information like coefficients as part of the inter-node cluster communications. The algorithms require that exchanged information stay synchronized across nodes. Secure this information through encryption in transit.

Implementation plan

- **Enable inter-node encryption in Amazon SageMaker**- In distributed computing environments, data transmitted between nodes can traverse wide networks, or even the internet. Enable inter-node encryption through the appropriate controls for the technology choices made. You can instruct SageMaker to automatically encrypt inter-container communication for your training job to ensure that data is passed over an encrypted tunnel.

- **Enable encryption in transit in Amazon EMR** - There are many applications and execution engines in the Hadoop ecosystem, providing a variety of tools to match the needs of your ML and analytics workloads. Amazon EMR has distributed cluster capabilities and is also an option for running training jobs on the data that is either stored locally on the cluster or in Amazon S3. Amazon EMR makes it easy to create and manage fully configured, elastic clusters of Amazon EC2 instances running Hadoop and other applications in the Hadoop ecosystem. Amazon EMR provides security configurations to set up data encryption at rest while stored on Amazon S3 and local Amazon EBS volumes. It also allows the set-up of Transport Layer Security (TLS) certificates for the encryption of data in transit.

Documents

- Protect Communications Between ML Compute Instances in a Distributed Training Job
- Amazon EMR Management guide - Security Data Protection & Encryption Options
- Apache Hadoop on Amazon EMR

Blogs

- Encrypt data in transit using a TLS custom certificate provider with Amazon EMR
- Secure Amazon EMR with Encryption

Examples

- Protect Communications Between ML Compute Instances in a Distributed Training Job
- TF Encrypted

MLSEC-10: Protect against data poisoning threats

Protect against data injection and data manipulation that pollutes the training dataset. Data injections can add corrupt training data that can result in incorrect model and outputs. Data manipulations can change existing data (for example, labels) that can result in inaccurate and weak predictive models. Identify and address corrupt data and inaccurate models using security methods and anomaly detection algorithms. Ensure immutability of datasets by providing protection against ransomware and malicious code in installed third-party packages.

Implementation plan

- **Use only trusted data sources for training data** - Verify that you have sufficient audit controls to replay activity and determine where a change occurred, by whom, and at what time. Before training, validate the quality of training data to look for strong outliers and potentially incorrect labels.
- **Look for underlying shifts in the patterns and distributions in training data** - Using monitoring of data drift, derive the impact to prediction variance. These skews can be an indicator of underlying data drift, and can provide an early warning of unauthorized access targeting the training data.
- **Identify model updates that negatively impact the results before moving them to production** - Determine if the retrained model results are different from the past model iteration. Use past test data and previous model iterations as a baseline.
- **Have a rollback plan** - Using versioned training data and versioned models, make sure you can revert to a known good working model in a failure scenario. Use a fully managed service to store features, such as the Amazon SageMaker Feature Store. See more details of the Amazon SageMaker Feature Store under the Reliability pillar section (MLREL-07).
- **Use low-entropy classification cases** - Look for significant, unexpected changes. Determine the bounds of thresholds, identify classifications that you do not expect to see, and alert if the retrained model exceeds them.

Documents

- Monitor Bias Drift for models in production
- Amazon SageMaker model registry now supports rollback of deployed models
- SageMaker Model Registry - Approve

Blogs

- Automated monitoring of your machine learning models with Amazon SageMakerModel Monitor and sending predictions to human review workflows using Amazon A2I
- Amazon SageMaker Model Monitor– Fully Managed Automatic Monitoring for Your Machine Learning Models
- 7 ways to improve security of your machine learning workflows

Videos

- AWS re:Invent 2020: Detect machine learning (ML) model drift in production

Examples

- Inawisdom: Machine Learning and Automated Model Retraining with SageMaker
- Amazon SageMaker Workshop- DevOps workflow

Reliability pillar – Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This section includes best practices to consider during model development that includes model training, tuning, and model performance evaluation.

Best practices

- MLREL-06: Enable CI/CD/CT automation with traceability
- MLREL-07: Ensure feature consistency across training and inference
- MLREL-08: Ensure model validation with relevant data
- MLREL-09: Establish data bias detection and mitigation

MLREL-06: Enable CI/CD/CT automation with traceability

Enable source code, data, and artifact version control of ML workloads to enable roll back to a specific version. Incorporate continuous integration (CI), continuous delivery (CD), and continuous training (CT) practices to ML workload operations. This will enable automation with added traceability.

Implementation plan

- **Use Amazon SageMaker Pipelines**- Manual changes to a system can cost additional time and impair reproducibility. Changes to an ML workload should be conducted, tracked and rolled back automatically. MLOps is a collection of best practices around integrating and deploying reproducible, auditable changes. MLOps increases your productivity while automating all facets of your ML development cycle (MLDC). Amazon SageMaker Pipelines is the first purpose-built,

continuous integration (CI), continuous delivery (CD), and continuous training (CT) service. With SageMaker Pipelines, create, automate, and manage end-to-end ML workflows at scale.

Documents

- AWS MLOps Framework
- SageMaker Pipelines Overview
- Continuous Delivery for Machine Learning on AWS

Blogs

- Improve your data science workflow with a multi-branch training MLOps pipeline using AWS
- Build a CI/CD pipeline for deploying custom machine learning models using AWS services
- Create Amazon SageMaker projects with image building CI/CD pipelines

Videos

- AWS re:Invent 2020: How to create fully automated ML workflows with Amazon SageMaker Pipelines
- Inawisdom: Machine Learning and Automated Model Retraining with SageMaker

Examples

- Amazon Sagemaker MLOps (with classic CI/CD tools) Workshop
- Amazon SageMaker secure MLOps

MLREL-07: Ensure feature consistency across training and inference

Ensure consistent, scalable, and highly available features between training and inference using a feature storage. This results in reducing the training-serving skew by keeping feature consistency between training and inference.

Implementation plan

- **Use Amazon SageMaker Feature Store** -Create, share, and manage features for ML development using SageMaker Feature Store. The Feature Store is a centralized store for features and associated metadata so features can be easily discovered and reused. The online store is used for low latency, real-time inference use cases. The offline store is used for training and batch inference. The Feature Store reduces the repetitive data processing and curation work required to convert raw data into features for training an ML algorithm. Features generated will be used for both training and inference, reducing the training-serving skew. The Feature Store enables feature consistency, feature standardization, and the ability to integrate with Amazon SageMaker Pipelines.

Documents

- Get started with Amazon SageMakerFeature Store

Blogs

- Store, Discover, and Share Machine Learning Features with Amazon SageMakerFeature Store
- Using streaming ingestion with Amazon SageMakerFeature Store to make ML-backed decisions in near-real time

Videos

- AWS re:Invent 2020: Amazon SageMakerFeature Store: Store, discover, & share features for ML apps
- Introducing Amazon SageMaker Feature Store - AWS re:Invent2020

Examples

- Amazon SageMaker Feature Store: Introduction to Feature Store
- Amazon SageMaker Feature Store
- Amazon SageMaker Feature Store Introduction
- Amazon SageMaker Feature Store: Streaming Aggregation

MLREL-08: Ensure model validation with relevant data

Put processes in place to include real and representative data for testing and validation. Data that does not include all possible patterns and scenarios will result in failures once model is in production. Check for a *distribution mismatch* between training, validation, and test data as well as the inference data.

Implementation plan

- **Use Amazon SageMaker Experiments** - Your models should be tested and validated using data that is representative of what they will encounter in production. This data can include both real-world data and engineered data. You should account for all scenarios in your training data so that you can avoid errors when your model is deployed to production. Use Amazon SageMaker Experiments to organize, track, compare, and evaluate your machine learning experiments.
- **Use Amazon SageMaker Model Monitor** - Consider implementing a plan to periodically test endpoints for deviations in model quality. Early detection of deviations can help you determine when to take corrective actions. SageMaker Model Monitor continually monitors the quality of Amazon SageMaker ML models in production. With Model Monitor, you can set alerts that notify you when there are deviations in the model quality.

Documents

- Evaluating ML Models
- Cross-Validation of Machine Learning Models

Blogs

- Test data quality at scale with Deequ
- Amazon SageMaker Model Monitor– Fully Managed Automatic Monitoring for Your Machine Learning Models

Examples

- Amazon SageMaker Model Monitor

MLREL-09: Establish data bias detection and mitigation

Detect and mitigate bias to avoid inaccurate model results. Establish bias detection methodologies at data preparation stage before training starts. Monitor, detect, and mitigate bias after the model is in production. Establish feedback loops to track the drift over time and initiate a re-training.

Implementation plan

- **Use Amazon SageMaker Clarify**- Amazon SageMaker Clarify helps improve your machine learning models by detecting potential bias and helping explain how these models make predictions. The fairness and explainability functionality provided by SageMaker Clarify takes a step towards enabling you to build trustworthy and understandable ML models. Clarify helps you with the following tasks:
  - Measure biases that can occur during each stage of the ML lifecycle. These stages include data collection, model training, model tuning, and model monitoring.
  - Generate model governance reports targeting risk and compliance teams and external regulators.
  - Provide explanations of the data, models, and monitoring used to assess predictions.

Documents

- Run SageMaker Clarify Processing Jobs for Bias Analysis and Explainability

Blogs

- Amazon SageMaker Clarify Detects Bias and Increases the Transparency of Machine Learning Models

Videos

- Introducing Amazon SageMakerClarify, part 1 - Bias detection - AWS re:Invent2020
- Introducing Amazon SageMakerClarify, part 2 - Model explainability - AWS re:Invent2020

Examples

- SageMaker Clarify

Performance efficiency pillar – Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve. This section includes best practices to consider during model development that includes model training, tuning and model performance evaluation.

Best practices

- MLPER-05: Optimize training and inference instance types
- MLPER-06: Explore alternatives for performance improvement
- MLPER-07: Establish a model performance evaluation pipeline
- MLPER-08: Establish feature statistics
- MLPER-09: Perform a performance trade-off analysis
- MLPER-10: Detect performance issues when using transfer learning

MLPER-05: Optimize training and inference instance types

Determine how the model type and data velocity affect the choice of training and inference instance types. Identify the right instance type that supports memory intensive training, or compute intensive training with high throughput and low latency real-time inference. The speed of model inferences is directly impacted by model complexity. Selection of high compute instances can accelerate inference speed. GPUs are often the preferred processor type to train many deep learning models. CPUs are often sufficient for the inference workloads.

Implementation plan

- **Experiment with alternative instance types to train and deploy** - Determine which instance types are most appropriate for your ML algorithm and use case. Use multiple instances for training for large datasets to take advantage of scale.

Documents

- Use Amazon SageMaker Notebook Instances
- Train a Model with Amazon SageMaker
- Distributed training libraries

Blogs

- Learn how to select ML instances on the fly in Amazon SageMaker Studio
- Optimizing I/O for GPU performance tuning of deep learning training in Amazon SageMaker
- Right-sizing resources and avoiding unnecessary costs in Amazon SageMaker

Videos

- How to choose the right instance type for ML inference
- The right instance type in Amazon SageMaker

Examples

- Amazon SageMaker End-to-End Example

MLPER-06: Explore alternatives for performance improvement

Perform benchmarks to improve the machine learning model performance. Benchmarking in ML involves evaluation and comparison of ML workloads with different algorithms, features, and architecture resources. It enables identifying the combination with optimal performance.

Options you can use when benchmarking include:

- Use more data to broaden the statistical range and improve the success metric of the model.
- Apply feature engineering to extract important signals in the data for the model.
- Make alternative algorithm selections for an optimal fit to the specifics of the data.
- Ensemble methods that combine the different advantages of multiple models.
- Tune the hyperparameters for a given algorithm to calibrate the model for the data.

Implementation plan

- **Use Amazon SageMaker Experiments to optimize algorithms and features** -Begin with a simple architecture, obvious features, and a simple algorithm to establish a baseline. Amazon SageMaker provides built-in algorithms for developing a baseline model. Use Amazon SageMaker Experiments to organize, track, compare, and evaluate your machine learning experiments. Test different algorithms with increasing complexity to observe performance. Combine models

into an ensemble to increase accuracy, but consider the potential loss of efficiency as a trade-off. Refine the features by selection and modify parameters to optimize model performance. Tune the model's hyperparameters to optimize performance using Amazon SageMaker Hyperparameter Optimization to automate the search.

Documents

- Improving Model Accuracy
- Evaluating ML Models
- Feature Processing with Spark ML and Scikit-learn
- Perform Automatic Model Tuning with SageMaker
- Amazon SageMaker Experiments: Track and Compare Tutorial

Blogs

- Running multiple HPO jobs in parallel on Amazon SageMaker
- Optimizing portfolio value with Amazon SageMaker automatic model tuning
- Utilizing XGBoost training reports to improve your models

Videos

- Tune your ML models to the highest accuracy with automatic model tuning
- Organize, Track, and Evaluate ML Training Runs With Amazon SageMaker Experiments

Examples

- Feature Engineering Immersion Day Workshop
- Improving Forecast Accuracy with Machine Learning
- Ensemble Predictions From Multiple Models

MLPER-07: Establish a model performance evaluation pipeline

Capture key metrics related to model performance using an end-to-end performance pipeline to evaluate the success of a model. Choose specific metrics based on the use case and the business

KPIs. Sample key metrics include training or validation errors, and prediction accuracy. Specific model performance metrics include Root Mean Squared Error (RMSE), accuracy, precision, recall, F1 score, and area under the curve (AUC). Establish a fully automated performance testing pipeline system to initiate evaluation every time there is an updated model or data.

Implementation plan

- **Create an end-to-end workflow with Amazon SageMaker Pipelines** - Start with a workflow template to establish an initial infrastructure for model training and deployment. SageMaker Pipelines helps you automate different steps of the ML workflow. These steps include data loading, data transformation, training, tuning, and deployment. With SageMaker Pipelines, you can share and reuse workflows to re-create or optimize models, helping you scale ML throughout your organization. Within SageMaker Pipelines, the SageMaker Model Registry tracks the model versions and respective artifacts. These artifacts include the metadata and lineage data collected throughout the model development lifecycle. SageMaker Model Registry can also enable automating model deployment with CI/CD.

Documents

- Define a Pipeline

Blogs

- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
- Extend Amazon SageMaker Pipelines to include custom steps using callback steps

Videos

- Introducing Amazon SageMaker Pipelines
- How to create fully automated ML workflows with Amazon SageMaker Pipelines

Examples

- SageMaker Pipelines – Immersion Day

MLPER-08: Establish feature statistics

Establish key statistics to measure changes in the data that affect model outcomes. The effect of changes in data on model inference depends on the sensitivity of the model to data features. Analyze the feature importance and sensitivity of the model to select the features to monitor. Monitor the statistics of features that have the largest influence on inferences. Place acceptability limits on the range of data to alert when important features drift outside the statistical range of the training data. Significant drifts in important features would suggest model re-training.

Implementation plan

- **Analyze and evaluate data** - Use Amazon SageMaker Data Wrangler to analyze the distribution of the selected features. After training the model, map out the regions in feature space where the predictions change abruptly and where the predictions are invariant. Establish a baseline for monitoring the data with Amazon SageMaker Model Monitor. Perform a sensitivity analysis of changes in the feature values near the decision boundaries of the model. Analyze the feature importance to understand how new data will affect the model's predictions. Amazon SageMakerExperiments will help to organize model testing. Use Amazon SageMaker Clarify to check for data biases and imbalances. Monitor the statistics of data used in production inferences. Consider retraining the model if the features are outside the original distribution of the training data.

Documents

- Amazon SageMaker Data Wrangler: Analyze and Visualize
- Detect Pretraining Data Bias

Blogs

- Exploratory data analysis, feature engineering, and operationalizing your data flow into your ML pipeline with Amazon SageMakerData Wrangler
- Amazon SageMaker Model Monitor– Fully Managed Automatic Monitoring for Your Machine Learning Models
- How Clarify helps machine learning developers detect unintended bias

Videos

- Prepare data for machine learning with ease, speed, and accuracy
- Detect machine learning (ML) model drift in production
- Accelerate data preparation with Amazon SageMakerData Wrangler

Examples

- Feature Engineering, ImmersionDay Workshop

MLPER-09: Perform a performance trade-off analysis

Perform alternative trade-off analysis to obtain optimal performance and accuracy for a given use-case data and business requirement.

- **Accuracy versus complexity trade-off**: The simpler a machine learning model is, the more explainable are its predictions. Deep learning predictions can potentially outperform linear regression or a decision tree algorithm, but at the cost of added complexity in interpretability and explainability.
- **Bias versus fairness trade-off**: Define a process for managing risks of bias and fairness in model performance. Business value most often aligns with models that have considered historical or sampling biases in the training data. Further consideration should be given to the disparate impact of inaccurate model predictions. For example, underrepresented groups are often more impacted by historical biases, which might perpetuate unfair practices.
- **Bias versus variance trade-off (supervised ML):** The goal is to achieve a trained model with the lowest bias versus variance tradeoff for a given data set. To help overcome bias and variance errors, you can use:
  - Cross validation
  - More data
  - Regularization
  - Simpler models
  - Dimension reduction (Principal Component Analysis)
  - Stop training early
- **Precision versus recall trade-off (supervised ML):** This analysis can be important when precision is more important than recall or vice versa. For example, optimization of precision is more

important when the goal is to reduce false positives. However, optimization of recall is more important when the goal is to reduce false negatives. It's not possible to have both high precision and high recall-if one is increased, the other decreases. A trade-off analysis helps identify the optimal option for analysis.

Implementation plan

Construct alternate workflows to optimize all aspects of business value - Complex models can deliver high accuracy. If the business requirements include low-latency, then the model might need to be simplified with lower complexity. Identify how trade-offs affect accuracy and the latency of inferences. Test these trade-offs using Amazon SageMaker Experiments to keep track of each model type. Amazon SageMakerClarify provides explanations of the data, models, and monitoring used to assess predictions. It can measure biases during each stage of the ML lifecycle. Provided explanations will help understanding how fairness affects the business use case. Complex ML models can be slower to return an inference and more difficult to deploy at the edge. Amazon SageMaker Neo enables developers to optimize ML models for inference on SageMaker in the cloud and supported devices at the edge.

Documents

- Evaluating ML Models
- AI Fairness and Explainability Whitepaper
- Optimize model performance using Amazon SageMaker Neo

Blogs

- Amazon SageMaker Experiments – Organize, Track And Compare Your Machine Learning Trainings
- Amazon SageMaker Clarify Detects Bias and Increases the Transparency of Machine Learning Models
- Unlock near 3x performance gains with XGBoost and Amazon SageMaker Neo

Videos

- Machine learning and society: Bias, fairness, and explainability

MLPER-10: Detect performance issues when using transfer learning

Monitor and ensure that the inherited prediction weights from a transferred model yield the desired results. This approach helps minimize the risk of weak learning and incorrect outputs using pre-trained models.

Implementation plan

- **Use Amazon SageMaker Debugger** - Transfer learning is a machine learning technique where a model pre-trained on one task is fine-tuned on a new task. When using the transfer learning approach, use Amazon SageMaker Debugger to detect hidden problems that might have serious consequences. Examine model predictions to see what mistakes were made. Validate the robustness of your model, and consider how much of this robustness is from the inherited capabilities. Validate input and preprocesses to the model for realistic expectations.

Blogs

- When does transfer learning work?
- Detecting hidden but non-trivial problems in transfer learning models using Amazon SageMaker Debugger

Cost optimization pillar – Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment. This section includes best practices to consider during model development.

Best practices

- MLCOST-09: Select optimal computing instance size
- MLCOST-10: Use managed build environments
- MLCOST-11: Select local training for small scale experiments
- MLCOST-12: Select an optimal ML framework
- MLCOST-13: Use automated machine learning

- MLCOST-14: Use managed training capabilities
- MLCOST-15: Use distributed training
- MLCOST-16: Stop resources when not in use
- MLCOST-17: Start training with small datasets
- MLCOST-18: Use warm-start and checkpointing hyperparameter tuning
- MLCOST-19: Use hyperparameter optimization technologies
- MLCOST-20 - Setup budget and use resource tagging to track costs
- MLCOST-21: Enable data and compute proximity
- MLCOST-22: Select optimal algorithms
- MLCOST-23: Enable debugging and logging

MLCOST-09: Select optimal computing instance size

Right size the training instances according to the ML algorithm used for maximum efficiency and cost reduction. Use debugging capabilities to understand the right resources to use during training. Simple models might not train faster on larger instances because they might not be able to benefit from additional compute resources. These models might even train slower due to the high GPU communication overhead. Start with smaller instances and scale as necessary.

Implementation plan

- **Use Amazon SageMaker Experiments** - Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Machine learning workloads can use either a CPU or a GPU instance. Select an instance type from the available EC2 instance types depending on the needs of your ML algorithm. Experiment with both CPU and GPU instances to learn which one gives you the best cost configuration. Amazon SageMaker lets you use a single instance or a distributed cluster of GPU instances. Use Amazon SageMaker Experiments to evaluate alternative options, and identify the size resulting in optimal outcome. With the pricing broken down by time and resources, you can optimize the cost of Amazon SageMaker and only pay for what is needed.
- **Use Amazon SageMaker Debugger** - Amazon SageMaker Debugger automatically monitors the utilization of system resources, such as GPUs, CPUs, network, and memory, and profiles your training jobs to collect detailed ML framework metrics. You can inspect all resource metrics visually through SageMaker Studio and take corrective actions if the resource is under-utilized to optimize cost.

Documents

- Amazon SageMaker Debugger

Blogs

- Right-sizing resources and avoiding unnecessary costs in Amazon SageMaker
- Identify bottlenecks, improve resource utilization, and reduce ML training costs with the deep profiling feature in Amazon SageMaker Debugger

Videos

- AWS re:Invent 2019: Choose the right instance type in Amazon SageMaker, with Texas Instruments

MLCOST-10: Use managed build environments

Consider using managed notebooks instead of local ones, or notebooks hosted on a server. Managed notebooks come bundled with security, network, storage, compute capabilities that take a lot of time and resources to develop locally. Managed ML build environment also makes it easy to decide the type of machine you prefer so you don't need to manage any complex AMIs or security groups-this makes it very easy to get started. It can also provide access to GPUs and big machines with large amounts of RAM that might not be possible on a local setup.

Implementation plan

- **Use Amazon SageMaker Notebooks** - An Amazon SageMaker notebook instance is a ML compute instance running the Jupyter Notebook App. SageMaker manages creating the instance and related resources. Use Jupyter notebooks in your notebook instance to prepare and process data, write code to train models, deploy models to SageMaker hosting, and test or validate your models. SageMaker provides hosted Jupyter notebooks that require no setup, so you can begin processing your training data sets immediately. With a few clicks in the SageMaker console, you can create a fully managed notebook instance, pre-loaded with useful libraries for machine learning. You only need to add your data.
- **Use Amazon SageMaker Studio** - Amazon SageMaker Studio provides a single, web-based visual interface where you can perform all ML development steps, improving data science team productivity. SageMaker Studio gives you complete access, control, and visibility into each step

required to build, train, and deploy models. You can quickly upload data, create new notebooks, train and tune models, move back and forth between steps to adjust experiments, compare results, and deploy models to production all in one place, making you much more productive. All ML development activities including notebooks, experiment management, automatic model creation, debugging, and model and data drift detection can be performed with SageMaker Studio.

- **Use SageMaker Canvas** - Amazon SageMaker Canvas, a new visual, no code capability that allows business analysts to build ML models and generate accurate predictions without writing code or requiring ML expertise. Its intuitive user interface lets you browse and access disparate data sources in the cloud or on premises, combine datasets with the click of a button, train accurate models, and then generate new predictions once new data is available.

Documents

- Amazon SageMaker Notebook Instances
- Amazon SageMaker Studio
- Amazon SageMaker Canvas

Blogs

- Amazon SageMaker Studio and SageMaker Notebook Instance now come with JupyterLab 3 notebooks to boost developer productivity
- Build, Share, Deploy: how business analysts and data scientists achieve faster time-to-market using no-code ML and Amazon SageMaker Canvas
- Amazon SageMaker Canvas – a Visual, No Code Machine Learning Capability for Business Analysts

Examples

- SageMaker Example Notebooks

MLCOST-11: Select local training for small scale experiments

Evaluate the requirements to train an ML model in the cloud versus on a local machine. Use local option when experimenting across different algorithms and configurations with small data sizes.

For large data, launch a cloud-based training cluster with one or more compute instances. Right size the compute instances in the training cluster based on the workload.

Implementation plan

- **Use Amazon SageMaker** - While experimenting with training a model with small datasets, use Amazon SageMaker notebook local mode. This will train your model on the notebook instance itself, instead of on a separate managed training cluster. You can iterate and test your work without having to wait for a new training or hosting cluster each time. This saves both time and cost associated with creating a managed training cluster. Experimentation can also occur outside of a notebook, for example on a local machine. From your local machine, you can use the SageMaker SDK to train and deploy models on AWS.

Blogs

- Use the Amazon SageMaker local mode to train on your notebook instance

Videos

- Train with Amazon SageMaker on your local machine

Examples

- Multiple examples showing how to run SageMaker in local mode

MLCOST-12: Select an optimal ML framework

Organize, track, compare and evaluate machine learning (ML) experiments and model versions. Identify the most cost-effective and optimal combination of instance types and ML frameworks. Examples of ML frameworks include TensorFlow, PyTorch, and Scikit-learn.

Implementation plan

- **Use Amazon SageMaker Experiments** - Amazon SageMaker Experiments lets you organize, track, compare, and evaluate your machine learning experiments. Using this service, you can experiment with various ML frameworks and see which one gives you the most cost-effective performance. AWS Deep Learning AMIs and AWS Deep Learning Containers enable you to use several open-source ML frameworks for training on your infrastructure. AWS Deep Learning

AMIs have popular deep learning frameworks and interfaces preinstalled including TensorFlow, PyTorch, Apache MXNet, Chainer, Gluon, Horovod, and Keras. The AMI or container can be launched on powerful infrastructure that has been optimized for ML performance. SageMaker also allows you to bring your own container where you can use any framework you choose.

Documents

- Use Machine Learning Frameworks, Python, and R with Amazon SageMaker

Blogs

- Right-sizing resources and avoiding unnecessary costs in Amazon SageMaker
- Amazon SageMaker Experiments – Organize, Track and Compare your Machine Learning Trainings

MLCOST-13: Use automated machine learning

Use automated data analyzer systems when building a model. These systems experiment with and select the best algorithm from the list of high-performing algorithms. They automatically test different solutions and parameter settings to achieve optimal models. The automated system speeds up the process, while eliminating manual experimentation and comparisons.

Implementation plan

- **Use Amazon SageMaker Autopilot** - Amazon SageMaker Autopilot automates key tasks of an automatic machine learning (AutoML) process. Autopilot explores your data, selects the algorithms relevant to your problem type, and prepares the data to facilitate model training and tuning. Autopilot applies a cross-validation resampling procedure automatically to all candidate algorithms. It tests the ability of these algorithms to predict using data they have not been trained on. It ranks all of the optimized models tested by their performance. Autopilot finds the best performing model that you can deploy at a fraction of the time normally required.

Blogs

- Amazon SageMaker Autopilot – Automatically Create High-Quality Machine Learning Models with Full Control and Visibility
- Customizing and reusing models generated by Amazon SageMakerAutopilot

Videos

- Automatically Build, Train, and Tune ML Models with Amazon SageMaker Autopilot
- Use Autopilot to automate and explore the machine learning process

Examples

- Customer Churn Prediction with Amazon SageMaker Autopilot
- SageMaker Autopilot

MLCOST-14: Use managed training capabilities

Machine learning model training can be an iterative, compute intensive, and time-consuming process. Instead of using the notebook itself, which might be running on a small instance, consider offloading the training to a managed cluster of compute resources including both CPUs and GPUs to train the model.

Implementation plan

- **Use Amazon SageMaker managed training capabilities** - Amazon SageMaker reduces the time and cost to train and tune ML models without the need to manage infrastructure. With SageMaker, easily train and tune ML models using built-in tools to manage and track training experiments, automatically choose optimal hyperparameters, debug training jobs, and monitor the utilization of system resources such as GPUs, CPUs, and network bandwidth. SageMaker can automatically scale infrastructure up or down based on your training job requirements, from one GPU to thousands, or from terabytes to petabytes of storage.SageMaker also offers the highest-performing ML compute infrastructure currently available-including Amazon EC2 P4d instances, which can reduce ML training costs by up to 60% compared with previous generations. And, since you pay only for what you use, you can manage your training costs more effectively.
- **Use the Amazon SageMaker Training Compiler** - To train deep learning (DL) models faster, you can use the Amazon SageMaker Training Compiler to accelerate the model training process by up to 50% through graph- and kernel-level optimizations that make more efficient use of GPUs. Moreover, you can add either data parallelism or model parallelism to your training script with a few lines of code, and the SageMaker distributed training libraries will automatically split models and training datasets across GPU instances to help you complete distributed training faster.
- **Use Amazon SageMaker managed Spot training** - Amazon SageMaker makes it easy to train machine learning models using managed Amazon EC2 Spot Instances. Managed Spot training

can optimize the cost of training models up to 90% over On-demand Instances. SageMaker manages the Spot interruptions on your behalf. You can specify which training jobs use Spot Instances and a stopping condition that specifies how long SageMaker waits for a job to run using Spot Instances. Metrics and logs generated during training runs are available in CloudWatch.

Documents

- Train a Model with Amazon SageMaker
- SageMaker Model Training
- Managed Spot Training in Amazon SageMaker
- Amazon SageMaker Training Compiler

Blogs

- Amazon SageMaker Simplifies Training Deep Learning Models with Billions of Parameters
- Choose the best data source for your Amazon SageMaker training job
- Managed Spot Training: Save Up to 90% On Your Amazon SageMaker Training Jobs
- Cinnamon AI saves 70% on ML model training costs with Amazon SageMaker Managed Spot Training

Examples

- Optimizing and Scaling Machine Learning Training

MLCOST-15: Use distributed training

Enable distributed training for a faster training time, when an algorithm allows it. Use multiple instances in a training cluster. Use managed services to help ensure all training instances are automatically shut down when training is completed.

Implementation plan

- **Use Amazon SageMaker Distributed training libraries** - The distributed training libraries in Amazon SageMaker automatically split large deep learning models and training datasets across AWS GPU instances in a fraction of the time it takes to do manually. SageMaker achieves these

efficiencies through two techniques: data parallelism and model parallelism. Model parallelism splits models too large to fit on a single GPU into smaller parts before distributing across multiple GPUs to train, and data parallelism splits large datasets to train concurrently to improve training speed.

Documents

- SageMaker's Distributed Data Parallel Library
- SageMaker's Distributed Model Parallel
- Distributed Training

Blogs

- New – Data Parallelism Library in Amazon SageMaker Simplifies Training on Large Datasets
- How Latent Space used the Amazon SageMaker model parallelism library to push the frontiers of large-scale transformers
- The science behind Amazon SageMaker's distributed-training engines
- Amazon SageMaker XGBoost now offers fully distributed GPU training

Videos

- AWS re:Invent 2020: Train billion-parameter models with model parallelism on Amazon SageMaker
- AWS re:Invent 2020: Fast training and near-linear scaling with Data Parallel in Amazon SageMaker

Examples

- Distributed Training
- Distributed training using Amazon SageMaker Distributed Data Parallel library and debugging using Amazon SageMaker Debugger
- SageMaker developer guide on distributed training

MLCOST-16: Stop resources when not in use

Stop resources that are not in use to reduce cost. For example, hosted Jupyter environments used to explore small samples of data, can be stopped when not actively in use. Where practical, commit the work, stop them, and restart when needed. The same approach can be used to stop the computing and the data storage services.

Implementation plan

- **Use CloudWatch Billing Alarms** - You can monitor your estimated AWS charges by using Amazon CloudWatch. When you enable the monitoring of estimated charges for your AWS account, the estimated charges are calculated and sent several times daily to CloudWatch as metric data. Use this feature to receive notifications when your resource charge exceeds a threshold amount.
- **Use SageMaker Lifecycle Configuration** - A lifecycle configuration provides shell scripts that run when you create the notebook instance or whenever you start one. When you create a notebook instance, you can create a new lifecycle configuration and its scripts or apply ones that you already have. Use a lifecycle configuration script to access AWS services from your notebook. These scripts enable checking notebook instance activities and shut them down if idle.
- **Use Amazon SageMaker Studio auto shutdown**- Amazon SageMaker Studio provides a unified, web-based visual interface for performing all ML development steps, making data science teams more productive. Idle SageMaker Studio notebooks can be detected and stopped using an auto-shutdown JupyterLab extension that can be installed manually or automatically. You can shut down individual resources, including notebooks, terminals, kernels, applications, and instances.

Documents

- Shut Down Resources

Blogs

- Save costs by automatically shutting down idle resources within Amazon SageMakerStudio

Examples

- Sagemaker-studio-auto-shutdown-extension

MLCOST-17: Start training with small datasets

Start experimentation with smaller datasets on a small compute instance or local system. This approach allows you to iterate quickly at low cost. After the experimentation period, scale up to train with the full dataset available on a separate compute cluster. Choose the appropriate storage layer for training data based on the performance requirements.

Implementation plan

- **Use SageMaker notebooks** - Notebooks are a popular way to explore and experiment with data in small quantities. Iterating with a small sample of the dataset locally and then scaling to train on the full dataset in a distributed manner is common in machine learning. Amazon SageMaker notebook instances provide a hosted Jupyter environment that can be used to explore small samples of data.

Documents

- Use Amazon SageMaker Notebook Instances
- Customize a Notebook Instance Using a Lifecycle Configuration Script

MLCOST-18: Use warm-start and checkpointing hyperparameter tuning

Where feasible, use warm start hyperparameter tuning. Warm start can consist of using a parent job for a model trained previously or using transfer learning. Warm start of hyperparameter tuning jobs eliminates the need to start a tuning job from scratch. Create a new hyperparameter tuning job that is based on selected parent jobs or pre-trained models. Use checkpointing capabilities to restart a training job from the last saved checkpoint. Reuse previous trainings as prior knowledge, or use checkpointing to accelerate the tuning process and reduce the cost.

Implementation plan

- **Use warm-start hyperparameter tuning** - Use warmstart to start a hyperparameter tuning job using one or more previous tuning jobs as a starting point. The results of previous tuning jobs are used to inform which combinations of hyperparameters to search over in the new tuning job. Hyperparameter tuning uses Bayesian or random search to choose combinations of hyperparameter values from ranges that you specify.
- **Use checkpointing hyperparameter tuning** - Use checkpoints in Amazon SageMaker to save the state of ML models during training. Checkpoints are snapshots of the model and can be

configured by the callback functions of ML frameworks. You can use the saved checkpoints to restart a training job from the last saved checkpoint.

Blogs

- Amazon SageMaker Automatic Model Tuning becomes more efficient with warm start of hyperparameter tuning jobs
- Running multiple HPO jobs in parallel on Amazon SageMaker

Videos

- Tune Your ML Models to the Highest Accuracy with Amazon SageMakerAutomatic Model Tuning

Examples

- Automatic Model Tuning : Warm Starting Tuning Jobs

MLCOST-19: Use hyperparameter optimization technologies

Use automatic hyperparameter tuning to run many training jobs and find the best version of your model. Use the algorithm and ranges of hyperparameters that you specify. Use appropriate hyperparameter ranges, as well as metrics that are realistic and meet the business requirements.

Implementation plan

- **Use SageMaker automatic model tuning** - SageMaker automatic model tuning, also known as hyperparameter tuning, finds the optimal model by running many training jobs on your dataset. It uses the algorithm and ranges of hyperparameters that you specify. It then chooses the hyperparameter values that result in a model that performs the best, as measured by a metric that you choose. To create a new hyperparameter optimization (HPO) tuning job for one or more algorithms, you need to define the settings for the tuning job. Create training job definitions for each algorithm being tuned, and configure the resources for the tuning job.

Documents

- Create a HPO tuning job

Blogs

- Running multiple HPO jobs in parallel on SageMaker

Videos

- Automatic model tuning with SageMaker

Examples

- Large HPO polling examples

MLCOST-20 - Setup budget and use resource tagging to track costs

If you need visibility of your ML cost, set up budgets and consider tagging your notebook instances. Examples of tags include the name of the project, the business unit, and environment (such as development, testing, or production).

Tags are useful for cost optimization and can provide a clear visibility into where money is being spent.

Implementation plan

- **Use AWS Budgets to keep track of cost** - AWS Budgets helps you track your Amazon SageMaker cost, including development, training, and hosting. You can also set alerts and get a notification when your cost or usage exceeds (or is forecasted to exceed) your budgeted amount. After you create your budget, you can track the progress on the AWS Budgets console.
- **Use AWS Cost Explorer** - Visualize, understand, and manage your AWS costs and usage over time using AWS Cost Explorer.
- **Tagging the resources** - Consider tagging your Amazon SageMaker notebook instances and the hosting endpoints. Cost allocation tags can help track and categorize your cost of ML. It can answer questions such as “Can I delete this resource to save cost?”

Documents

- Managing your costs with AWS Budgets
- Using Cost Allocation Tags

- Tagging Best Practices

Blogs

- Optimizing costs for machine learning with Amazon SageMaker
- Automate Cost Control using Service Catalog and AWS Budgets

MLCOST-21: Enable data and compute proximity

Ensure that the Region used for training and developing models is the same as the one used for data. This approach helps minimize the time and cost of transferring data to the computation environment.

Implementation plan

- **Keep data and compute resources in close proximity** - Amazon EC2 is hosted in multiple locations world-wide. These locations are composed of Regions, Availability Zones, Local Zones, AWS Outposts, and Wavelength Zones. Each *Region* is a separate geographic area. If you are launching a compute cluster, you should launch the cluster in close proximity to your data to get the best performance. Say, your Amazon S3 bucket is in the US West (Oregon) Region, you should launch your cluster in the US West (Oregon) Region to avoid Cross-Region data transfer fees.

Documents

- Regions and Zones

Blogs

- Overview of Data Transfer Costs for Common Architectures

Videos

- AWS re:Invent 2018: Architecture Patterns for Multi-Region Active-Active Applications

MLCOST-22: Select optimal algorithms

Identify the basic machine learning paradigm that addresses your ML problem type. Basic machine learning paradigms include: supervised learning, unsupervised learning and reinforcement learning. Identify the acceptable level of tradeoff between explainability and success metrics per business requirements. Run prototypes and experiments to explore high performing algorithms. Select the optimal cost-efficient algorithms that meet all the business requirements. Improved runtime performance of a tuned algorithm within business requirements, is one step towards optimizing the cost of ML.

Implementation plan

- Adopt optimal practices
  - Start with simple algorithms, such as regression, and work towards more complex algorithms, such as deep learning, to compare the accuracy of the models. Optimize hyperparameters to determine which algorithm yields the best metrics for the business use case.
  - When selecting the optimal algorithm, run trade-off analysis between data constraints, computational performance, and maintenance efforts. For example, deep learning networks might produce more accurate results, but require more data than tree-based methods. Deep learning methods are also more difficult to maintain.
- Use AWS services
  - Use Amazon SageMaker with a suite of built-in algorithms to train and deploy machine learning models. AWS provides optimized versions of frameworks, such as TensorFlow, Chainer, Keras, and Theano. These frameworks include optimizations for high-performance training across Amazon EC2 instance families.
  - Use Amazon SageMaker Experiments to keep track of the models during testing.
  - Use Amazon SageMaker Autopilot to select algorithms automatically.
  - Discover pre-trained ML models on AWS Marketplace - Pre-trained ML models are ready-to-use models that can be quickly deployed on *Amazon SageMaker*. By pre-training the models for you, solutions in AWS Marketplace take care of the heavy lifting, helping your team deliver ML powered features faster and at a lower cost.

Documents

- Choose an Algorithm
- Manage Machine Learning with Amazon SageMakerExperiments

- Automate model development with Amazon SageMakerAutopilot

Blogs

- Optimizing costs for machine learning with Amazon SageMaker
- Amazon SageMaker Experiments – Organize, Track, and Compare Your Machine Learning Trainings
- Amazon SageMaker Autopilot – Automatically Create High-Quality Machine Learning Models with Full Control and Visibility
- Streamline modeling with Amazon SageMaker Studio and the Amazon Experiments SDK

Videos

- Accelerate Machine Learning Projects with Hundreds of Algorithms and Models in AWS Marketplace
- Organize, Track, and Evaluate ML Training Runs with Amazon SageMaker Experiments

MLCOST-23: Enable debugging and logging

Ensure that there are sufficient logs and metrics recorded to capture the runtime and resource consumption. The collected logs and metrics can be analyzed to identify the areas for improvement. Monitor compute and data storage consumption. Instrument the machine learning code, and use debugging tools to capture metrics at runtime.

Implementation plan

- **Use Amazon SageMaker Debugger** - Amazon SageMaker Debugger captures the state of a training job at periodic intervals. It provides visibility into the ML training process by monitoring, recording, and analyzing data with the ability to perform interactive exploration of data captured during training. The debugger has an alerting capability for errors detected during training. For example, it can automatically detect and alert you to commonly occurring errors, such as gradient values getting too large or too small.
- **Use Amazon CloudWatch** -Logs generated during training by Amazon SageMaker are logged to Amazon CloudWatch Logs. Use an AWS KMS key to encrypt log data ingested by Amazon CloudWatch Logs.

Documents

- Logging and Monitoring
- Logging Amazon ML API Calls with AWS CloudTrail
- Amazon SageMaker Debugger - Setup and Use

Blogs

- Build Your Own Log Analytics Solution on AWS
- Profile Your Machine Learning Training Jobs with Amazon SageMaker Debugger
- Amazon SageMaker Debugger – Debug Your Machine Learning Models
- ML Explainability with Amazon SageMaker Debugger
- The science behind SageMaker's cost-saving Debugger

Examples

- Debugger example notebooks

Video

- Train ML models faster with better insights using Amazon SageMaker Debugger
- Debugging a Customer Churn Model Using SageMaker Debugger

Sustainability pillar – Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. This section includes best practices to consider while developing models that includes training, tuning, and model performance evaluation.

Best practices

- MLSUS-07: Define sustainable performance criteria
- MLSUS-08: Select energy-efficient algorithms
- MLSUS-09: Archive or delete unnecessary training artifacts
- MLSUS-10: Use efficient model tuning methods

Related best practices

- **Tradeoff analysis on custom versus pre-trained models** (MLCOST-04) - Consider whether the workload needs to be developed as a custom model. Many workloads can use the managed AWS AI services. Using these services means that you won't need the associated resources to collect, store, and process data and to prepare, train, tune, and deploy an ML model. If adopting a fully managed AI service is not appropriate, evaluate if you can use pre-existing datasets, algorithms, or models. AWS Marketplace offers over 1,400 ML-related assets that customers can subscribe to. You can also fine-tune an existing model starting from a pre-trained model, like those available on Hugging Face or SageMaker JumpStart. Using pre-trained models from third parties can reduce the resources you need for data preparation and model training.
- **Enable debugging and logging** (MLCOST-23) - A debugger like SageMaker Debugger can identify training problems like system bottlenecks, overfitting and saturated activation functions. It provides built-in rules like LowGPUUtilization or Overfit to monitor your workload and automatically stop a training job as soon as it detects an issue (such as bug, job failing to converge…). SageMaker Debugger also provides profiler capabilities to detect under-utilization of system resources and help right-size your environment. This helps avoid unnecessary carbon emissions.
- **Select optimal computing instance size** (MLCOST-09) - Use SageMaker Studio to switch instance types on the fly based on your needs (for example, use a low power type for exploratory data analysis, and then switch to GPU only to prototype some neural network code). Right size your training jobs with Amazon CloudWatch metrics that monitor resources, such as CPU, GPU, memory, and disk utilization.
- **Select local training for small scale experiments** (MLCOST-11) and **Start training with small datasets** (MLCOST-17) - Experiment with smaller datasets in your development notebook. This approach allows you to iterate quickly with limited carbon emission.
- **Stop resources when not in use** (MLCOST-16) - When building your model, use lifecycle configuration scripts to automatically stop idle SageMaker Notebook instances. If you are using SageMaker Studio, install the auto-shutdown Jupyter extension to detect and stop idle resources. Use the fully managed training process provided by SageMaker to automatically launch training instances and shut them down as soon as the training job is complete. This minimizes idle compute resources and thus limits the environmental impact of your training job.

MLSUS-07: Define sustainable performance criteria

Make trade-offs between your model's accuracy and its environmental impacts. When we focus only on the model's accuracy, we "ignore the economic, environmental, or social cost of reaching the reported accuracy." Because the relationship between model accuracy and complexity is at best logarithmic, training a model longer or looking for better hyperparameters only leads to a small increase in performance.

Implementation plan

- **Establish sustainable performance criteria** - Define performance criteria that support your sustainability goals while meeting your business requirements, but not exceeding them.
- **Make trade-offs** - Acceptable decreases in model performance can significantly reduce sustainability impacts of your models.
- **Stop training early** - In Automatic Model Tuning, early stopping stops the training jobs that a hyperparameter tuning job launches early when they are not improving significantly as measured by the objective metric. Similarly, SageMaker Debugger provides rules to automatically stop a training job as soon as it detects an issue (such as bug, job failing to converge...).

Documents

- Automatic Model Tuning with SageMaker - Stop Training Jobs Early
- Amazon SageMaker Debugger - Built-in Rules: LossNotDecreasing

Blogs

- Optimize AI/ML workloads for sustainability: Part 2, model development
- Amazon SageMaker Automatic Model Tuning now supports early stopping of training jobs

Metrics

- Track the metrics related to the resources provisioned for your training jobs (InstanceCount, InstanceType, and VolumeSizeInGB)
- Measure the efficient use of these resources (CPUUtilization, GPUUtilization, GPUMemoryUtilization, MemoryUtilization, and DiskUtilization) in the SageMaker Console, the CloudWatch Console or your SageMaker Debugger Profiling Report

MLSUS-08: Select energy-efficient algorithms

To minimize resource usage, replace algorithms with more efficient versions that produce the same result.

Implementation plan

- **Begin with a simple algorithm to establish a baseline** - Then test different algorithms with increasing complexity to observe whether performance has improved. If so, compare the performance gain against the difference in resources required.
- **Try to find simplified versions of algorithms** - This approach helps you use less resources to achieve a similar outcome. For example, DistilBERT, a distilled version of BERT, has 40% fewer parameters, runs 60% faster, and preserves 97% of its performance.
- **Compress models size without significant loss of accuracy** - Use pruning to remove weights that don't contribute much to the model. Use quantization to represent numbers with the low-bit integers without incurring significant loss in accuracy. These techniques speed up inference and save energy with limited impact on accuracy.
- **Employ Amazon SageMaker Neo** - Optimize ML models for inference on SageMaker in the cloud and supported devices at the edge.

Documents

- Explore alternatives for performance improvement
- DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter
- Optimize model performance using Amazon SageMaker Neo

Blogs

- Optimize AI/ML workloads for sustainability: Part 2, model development
- Pruning machine learning models with Amazon SageMaker Debugger and Amazon SageMaker Experiments
- Reduce ML inference costs on Amazon SageMaker with hardware and software acceleration
- Unlock near 3x performance gains with XGBoost and Amazon SageMaker Neo

Metrics

- Track the metrics related to the resources provisioned for your training and inference jobs (InstanceCount, InstanceType, and VolumeSizeInGB) and the efficient use of these resources (CPUUtilization, GPUUtilization, GPUMemoryUtilization, MemoryUtilization, and DiskUtilization) in the SageMaker Console, the CloudWatch Console or your SageMaker Debugger Profiling Report

MLSUS-09: Archive or delete unnecessary training artifacts

Remove training artifacts that are unused and no longer required to limit wasted resources. Determine when you can archive training artifacts to more energy-efficient storage or safely delete them.

Implementation plan

- **Clean up unneeded training resources** - Organize your ML experiments with SageMaker Experiments to clean up training resources you no longer need.
- **Reduce the volume of logs you keep** - By default, CloudWatch retains logs indefinitely. By setting limited retention time for your notebooks and training logs, you'll avoid the environmental impact of unnecessary log storage.

Documents

- Clean Up Amazon SageMaker Experiment Resources
- Change log data retention in CloudWatch Logs

Blogs

- Optimize AI/ML workloads for sustainability: Part 2, model development
- Clean up Your Container Images with Amazon ECR Lifecycle Policies

Metrics

- Measure and optimize the total size of your Amazon S3 buckets and storage class distribution, using Amazon S3 Storage Lens
- Measure and optimize the size your of CloudWatch log groups

MLSUS-10: Use efficient model tuning methods

Implement an efficient strategy to optimize hyperparameter values to minimize the resources required to complete model training. Avoid a brute force strategy wherever possible, as it tests hyperparameter values without concern for the number of resources used.

Implementation plan

- **Adopt sustainable tuning job strategy** - Prefer Hyperband or Bayesian search over random search (and avoid grid search). Bayesian search makes intelligent guesses about the next set of parameters to pick based on the prior set of trials. It typically requires 10 times fewer jobs than random search, and thus 10 times less compute resources, to find the best hyperparameters. SageMaker Automatic Model Tuning now supports Hyperband, a new search strategy that can find the optimal set of hyperparameters up to three times faster than Bayesian search for large-scale models such as deep neural networks that address computer vision problems.
- **Limit the maximum number of concurrent training jobs** - Running hyperparameter tuning jobs concurrently gets more work done quickly. However, with the Bayesian optimization strategy, a tuning job improves only through successive rounds of experiments. Typically, running one training job at a time achieves the best results with the least amount of compute resources.
- **Carefully choose the number of hyperparameters and their ranges** - You get better results and use less compute resources by limiting your search to a few parameters and small ranges of values. If you know that a hyperparameter is log-scaled, convert it to further improve the optimization.

Documents

- Perform Automatic Model Tuning with SageMaker
- Choosing the Best Number of Concurrent Training Jobs
- Choosing Hyperparameter Ranges
- Amazon SageMaker Automatic Model Tuning now provides up to 3x faster hyperparameter tuning with Hyperband as a new search strategy

Blogs

- Optimize AI/ML workloads for sustainability: Part 2, model development
- Amazon SageMaker automatic model tuning produces better models, faster

- Amazon SageMaker Automatic Model Tuning: Using Machine Learning for Machine Learning
- Amazon SageMaker Automatic Model Tuning now provides up to three times faster hyperparameter tuning with Hyperband

Metrics

- Track the metrics related to the resources provisioned for your hyperparameter tuning jobs (InstanceCount, InstanceType, and VolumeSizeInGB)
- Measure the efficient use of these resources (CPUUtilization, GPUUtilization, GPUMemoryUtilization, MemoryUtilization, and DiskUtilization) in the SageMaker Console and the CloudWatch Console

ML lifecycle phase - Deployment

After you have trained, tuned, and evaluated your model, you can deploy it into production and make predictions against this deployed model. Amazon SageMaker Studio can convert notebook code to production-ready jobs without the need to manage the underlying infrastructure. Be sure to use a governance process. Controlling deployments through automation combined with manual or automated quality gates ensures that changes can be effectively validated with dependent systems prior to deployment to production.

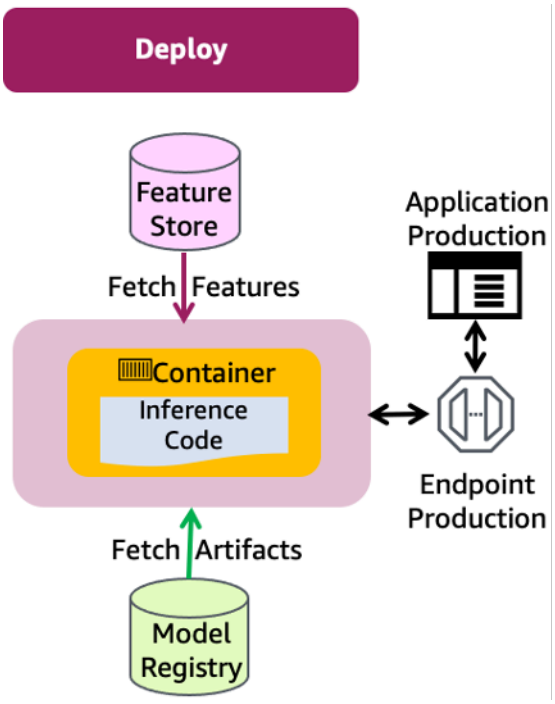

Figure 15 – Deployment architecture diagram

Figure 15 illustrates the deployment phase of the ML lifecycle in production. An application sends request payloads to a production endpoint to make inference against the model. Model artifacts are fetched from the model registry, features are retrieved from the feature store, and the inference code container is obtained from the container repository.

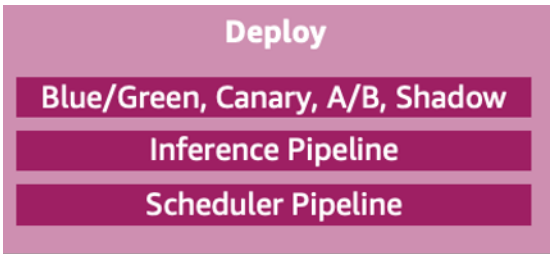

Figure 16: Deployment main components

Figure 16 lists key components of production deployment including:

- **Blue/green, canary, A/B, shadow deployment/testing** - Deployment and testing strategies that reduce downtime and risks when releasing a new or updated version.
  - The **blue/green** deployment technique provides two identical production environments (initially **blue** is the existing infrastructure and **green** is an identical infrastructure for testing). Once testing is done on the **green** environment, live application traffic is directed to it from the **blue** environment. Then the roles of the blue/green environments are switched.
  - With a **canary** deployment, a new release is deployed to a small group of users while other users continue to use the previous version. Once you're satisfied with the new release, you can gradually roll it out to all users.
  - **A/B** testing strategy enables deploying changes to a model. Direct a defined portion of traffic to the new model. Direct the remaining traffic to the old model. A/B testing is similar to canary testing, but has larger user groups and a longer time scale, typically days or even weeks.
  - With **shadow** deployment strategy, the new version is available alongside the old version. The input data is run through both versions. The older version is used for servicing the production application and the new one is used for testing and analysis.
- **Inference pipeline** - Figure 17 shows the inference pipeline that automates capturing of the prepared data, performing predictions and post-processing for real-time or batch inferences.
- **Scheduler pipeline** - Deployed model is representative of the latest data patterns. When configured as shown in Figure 17, re-training at intervals can minimize the risk of data and

concept drifts. A scheduler can initiate a re-training at business defined intervals. Data preparation, CI/CD/CT, and feature pipelines will also be active during this process.

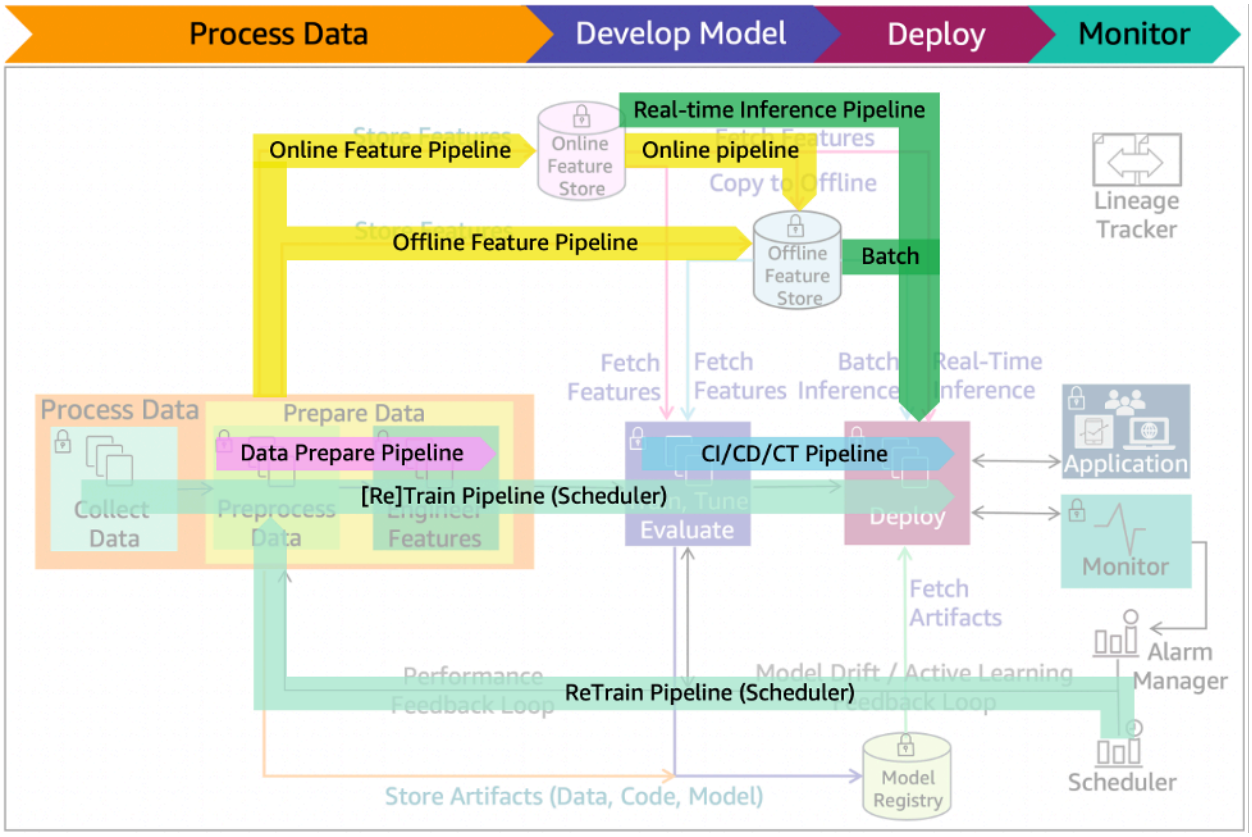

Figure 17: ML lifecycle with scheduler re-train, and batch/real-time inference pipelines

Best practices

- Operational excellence pillar – Best practices
- Security pillar - Best practices
- Reliability pillar – Best practices
- Performance efficiency pillar – Best practices
- Cost optimization pillar - Best practices
- Sustainability pillar - Best practices

Operational excellence pillar – Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures.

Best practices

- MLOE-14: Establish deployment environment metrics

MLOE-14: Establish deployment environment metrics

Measure machine learning operations metrics to determine the performance of a deployed environment. These metrics include memory and CPU/GPU usage, disk utilization, ML endpoint invocations, and latency.

Implementation plan

- **Record performance-related metrics** - Use a monitoring and observability service to record performance-related metrics. These metrics can include database transactions, slow queries, I/O latency, HTTP request throughput, service latency, and other key data.
- **Analyze metrics when events or incidents occur** - Use monitoring dashboards and reports to understand and diagnose the impact of an event or incident. These views provide insight into what portions of the workload are not performing as expected.
- **Establish key performance indicators (KPIs) to measure workload performance** - Identify the KPIs that indicate whether the workload is performing as intended. An API-based workload might use overall response latency as an indication of overall performance, while an e-commerce site might choose to use the number of purchases as its KPI.
- **Use monitoring to generate alarm-based notifications** - Monitor metrics for the defined KPIs and generate alarms automatically when the measurements are outside expected boundaries.
- **Review metrics at regular intervals** - As routine maintenance, or in response to events or incidents, review what metrics are collected and identify the metrics that were key in addressing issues. Identify any additional metrics that would help to identify, address, or prevent issues.
- **Monitor and alarm proactively** - Use KPIs, combined with monitoring and alerting systems, to proactively address performance-related issues. Use alarms to initiate automated actions to remediate issues where possible. Escalate the alarm to those able to respond if an automated response is not possible. Use a system to predict expected KPI values, and generate alerts and automatically halt or roll back deployments if KPIs are outside of the expected values.
- **Use Amazon CloudWatch** - Use Amazon CloudWatch metrics for SageMaker endpoints to determine the memory, CPU usage, and disk utilization. Set up CloudWatch Dashboards to visualize the environment metrics and establish CloudWatch alarms to initiate a notification via Amazon SNS (Email, SMS, WebHook) to notify on events occurring in the runtime environment.

- **Use Amazon EventBridge** - Consider defining an automated workflow using Amazon EventBridge to respond automatically to events. These events can include training job status changes, endpoint status changes, and increasing the compute environment capacity after it crosses a defined threshold (such as CPU or disk utilization).
- **Use AWS Application Cost Profiler** - Use AWS Application Cost Profiler to report the cost per tenant (model/user).

Documents

- DevOps and AWS
- Next Generation SageMaker Notebooks – Now with Built-in Data Preparation, Real-Time Collaboration, and Notebook Automation

Videos

- DevOps at Amazon: A Look at Our Tools and Processes

Security pillar - Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security.

Best practices

- MLSEC-11: Protect against adversarial and malicious activities

MLSEC-11: Protect against adversarial and malicious activities

Add protection inside and outside of the deployed code to detect malicious inputs that might result in incorrect predictions. Automatically detect unauthorized changes by examining the inputs in detail. Repair and validate the inputs before they are added back to the pool.

Implementation plan

- **Evaluate the robustness of the algorithm** - Evaluate your use case and determine bad predictions or classifications. Use sensitivity analysis to evaluate the robustness of the algorithm against increasingly perturbed inputs to understand susceptibility to manipulated inputs.

- **Build for robustness from the start** - Select diverse features to improve the algorithm's ability to handle outliers. Consider using models in an ensemble for increased diversity in decisions and for robustness around decision points.
- **Identify repeats** - Detect similar repeated inputs to the model to indicate possible threats to the decision boundaries using Amazon SageMaker Model Monitor to run a SageMaker processing job on a periodic interval to analyze the inference data. This can take the form of model brute forcing, where threats iterate only a limited set of variables to determine what influences decision points and derive feature importance.
- **Lineage tracking** - If retraining on untrusted or unvalidated inputs, make sure any model skew is traced back to the data and pruned before retraining a replacement model.
- **Use secure inference API endpoints** - Host the model so that a consumer of the model can perform inference against it securely. Permit consumers using the API to define the relationship, restrict access to the base model, and provide monitoring of model interactions.

Documents

- Deep ensembles
- Empirical demonstration of deterministic overconfidence
- Making Machine Learning Robust Against Adversarial Inputs

Blogs

- 7 ways to improve security of your machine learning workflows
- Run ensemble ML models on Amazon SageMaker

Videos

- Security and Privacy of Machine Learning

Examples

- Evasion Attacks against Banking Fraud Detection Systems
- Adversarial Robustness Libraries
- SecML: A library for Secure and Explainable Machine Learning

Reliability pillar – Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to.

Best practices

- MLREL-10: Automate endpoint changes through a pipeline
- MLREL-11: Use an appropriate deployment and testing strategy

MLREL-10: Automate endpoint changes through a pipeline

Manual change management is error prone, and incurs a high effort cost. Use automated pipelines (that integrate with a change management tracking system) to deploy changes to your model endpoints. Versioned pipeline inputs and artifacts allow you to track the changes and automatically rollback after a failed change.

Implementation plan

- **Use Amazon SageMaker Pipelines** - Deploying changes through a pipeline is a safe engineering method that enables consistency. Amazon SageMaker Pipelines is the purpose-built, easy-to-use continuous integration and continuous delivery (CI/CD) service which enables you to create, automate, and manage end-to-end ML workflows at scale.

Documents

- SageMaker Pipelines Overview
- What is a SageMaker Project?

Blogs

- Build a CI/CD pipeline for deploying custom machine learning models using AWS services
- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
- Build Custom SageMaker Project Templates – Best Practices

Videos

- AWS re:Invent 2021: Implementing MLOps practices with Amazon SageMaker

- AWS re:Invent 2020: How to create fully automated ML workflows with Amazon SageMaker Pipelines
- AWS on Air 2020: AWS What's Next ft. Amazon SageMaker Pipelines

Examples

- Comparing model metrics with SageMaker Pipelines and SageMaker Model Registry

MLREL-11: Use an appropriate deployment and testing strategy

Run a trade-off analysis across available and relevant deployment/testing strategies (such as blue/green, canary, shadow, and A/B testing) and select the one that meets your business requirements.

Implement metrics that evaluate model performance to identify when a rollback or roll-forward is required. When architecting for rollback or roll-forward, evaluate the following for each model:

- Where is the model artifact stored?
- Are model artifacts versioned?
- What changes are included in each version?
- What version of the model is deployed for a deployed endpoint?

Implementation plan

- **Deployment/testing in Amazon SageMaker:** SageMaker provides managed deployment strategies for testing new versions of your models in production.
  - See the explanation associated with Figure 16 for details of **blue/green, canary, and A/B deployment/testing.**
  - **Blue/green deployments using Amazon SageMaker:** Amazon SageMaker automatically uses a blue/green deployment to maximize the availability of your endpoints when updating a SageMaker real-time endpoint. The various traffic shifting modes in blue/green deployment give you more granular control over shifting traffic between the blue and green fleet. For more details, see Blue/Green deployments in SageMaker.
  - **Canary deployment using Amazon SageMaker:** The canary deployment option lets you shift one small portion of your traffic (a canary) to the green fleet and monitor it for a baking period. If the canary succeeds on the green fleet, the rest of the traffic is shifted from the blue fleet to the green fleet before stopping the blue fleet.

For more information, review canary traffic shifting in SageMaker.

- **Linear deployment using Amazon SageMaker:** Linear traffic shifting allows you to gradually shift traffic from your old fleet (blue fleet) to your new fleet (green fleet). With linear traffic shifting, you can shift traffic in multiple steps, minimizing the chance of a disruption to your endpoint. This blue/green deployment option gives you the most granular control over traffic shifting.
- **A/B testing using Amazon SageMaker:** Performing A/B testing between a new model and an old model with production traffic can be an effective final step in the validation process for a new model. In A/B testing, you test different variants of your models and compare how each variant performs. If the newer version of the model delivers better performance than the previously-existing version, replace the old version of the model with the new version in production. For more details, review test models in production in the SageMaker documentation.

Documents

- Deployment guardrails: a set of model deployment options in Amazon SageMaker Inference to update your machine learning models in production
- Blue/Green deployments in Amazon SageMaker
- Blue/Green Deployment on AWS
- Perform a canary-based deployment using the blue/green strategy and AWS Lambda
- Amazon SageMaker – Testing models in production – Model A/B test example

Blogs

- Take advantage of advanced deployment strategies using Amazon SageMaker deployment guardrails
- A/B Testing ML models in production using Amazon SageMaker
- Dynamic A/B testing for machine learning models with Amazon SageMaker MLOps projects
- Deploy shadow ML models in Amazon SageMaker
- Safely deploying and monitoring Amazon SageMaker endpoints with AWS CodePipeline and AWS CodeDeploy

Videos

- AWS re:Invent 2020: Canaries in the code mines: Monitoring deployment pipelines

Examples

- Amazon SageMaker Inference Deployment Guardrails
- Amazon SageMaker A/B Testing Pipeline
- Amazon SageMaker Safe Deployment Pipeline
- ML Model Shadow Deployment Strategy on AWS

Performance efficiency pillar – Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve.

Best practices

- MLPER-11: Evaluate cloud versus edge options for machine learning deployment
- MLPER-12: Choose an optimal deployment option in the cloud

MLPER-11: Evaluate cloud versus edge options for machine learning deployment

Evaluate if machine learning applications require near-instantaneous inference results or require inference without network connectivity. Offering the lowest latency possible might require the removal of costly roundtrips to the nearest API endpoints. A reduction in latency can be achieved by running the inference directly on the device itself (on the *edge*). A common use-case for such a requirement is predictive maintenance in factories.

Implementation plan

- **Optimize model deployment on the edge** - Training and optimizing machine learning models require massive computing resources, so it is a natural fit for the cloud. Inference takes a lot less computing power and is often done in real time when new data is available. When getting inference results with very low latency, confirm that your IoT applications can respond quickly to local events. Evaluate and choose the option to meet your business requirements.
  - Amazon SageMaker Edge enables machine learning on edge devices by optimizing, securing, and deploying models to the edge, and then monitoring these models on your fleet of devices,

such as smart cameras, robots, and other smart-electronics, to reduce ongoing operational costs. Customers who train models in TensorFlow, MXNet, PyTorch, XGBoost, and TensorFlow Lite can use SageMaker Edge to improve their performance, deploy them on edge devices, and monitor their health throughout their lifecycle. SageMaker Edge Compiler optimizes the trained model to be run on an edge device. SageMaker Edge Agent allows you to run multiple models on the same device. The Agent collects prediction data based on the logic that you control, such as intervals, and uploads it to the cloud so that you can periodically retrain your models over time. SageMaker Edge cryptographically signs your models so you can verify that they were not tampered with as they move from the cloud to edge devices.

- Amazon SageMaker Neo enables ML models to be trained once and then run anywhere in the cloud and at the edge. SageMaker Neo consists of a compiler and a runtime. The compilation API reads models exported from various frameworks, converts them into framework-agnostic representations, and generates optimized binary code (to run faster with no loss in accuracy). The compiler uses a machine learning model to apply the performance optimizations that extract the best available performance for your model on the cloud instance or edge device. The runtime for each target platform then loads and runs the compiled model.
- SageMaker Neo optimizes machine learning models for inference on cloud instances and edge devices. SageMaker Neo optimizes the trained model and compiles it into an executable. You then deploy the model as a SageMaker endpoint or on supported edge devices and start making predictions.
- AWS IoT Greengrass enables ML inferences on edge devices using models trained in the cloud. These models can be built using Amazon SageMaker, AWS Deep Learning AMI, or AWS Deep Learning Containers. These models can be stored in Amazon S3 before being deployed on edge devices.

Documents

- AWS IoT Greengrass ML Inference
- Amazon SageMaker Edge Manager
- Getting Started with Neo on Edge Devices

Blogs

- Machine Learning at the Edge: Using and Retraining Image Classification Models with AWS IoT Greengrass

- Monitor and Manage Anomaly Detection Models on a fleet of Wind Turbines with Amazon SageMaker Edge Manager
- SageMaker Edge Manager Simplifies Operating Machine Learning Models on Edge Devices
- Amazon SageMaker Neo Helps Detect Objects and Classify Images on Edge Devices
- Machine Learning at the Edge with AWS Outposts and Amazon SageMaker

Videos

- Machine Learning at the Edge
- Getting Started Using Machine Learning at the Edge
- Train ML Models Once, Run Anywhere in the Cloud & at the Edge with Amazon SageMaker Neo

MLPER-12: Choose an optimal deployment option in the cloud

If models are suitable for cloud deployment, you should determine how to deploy them for best performance efficiency according to frequency, latency, and runtime requirements in your use cases.

Implementation plan

- **Amazon SageMaker Real-time Inference** - Use if you need a persistent endpoint for near-instantaneous response from the ML model for requests that can come in any time. You can host the models behind an HTTPS endpoint to be integrated with your applications. SageMaker real-time endpoints are fully managed and support autoscaling.
- **Amazon SageMaker Serverless Inference** - Use if you receive spiky inference requests that vary substantially in rate and volume. This is a purpose-built inference option that makes it easy to deploy and scale ML models without managing any servers. Serverless Inference is ideal for workloads which have idle periods between traffic spurts and can tolerate cold starts.
- **Amazon SageMaker Asynchronous Inference** - Use if you have model requests with large payload sizes (up to 1GB), long processing times (up to 15 minutes), and near-instantaneous latency requirements, SageMaker Asynchronous Inference is ideal as it has larger payload limit and longer time-out limit compared to SageMaker Real-time inference. SageMaker Asynchronous Inference queues incoming requests and processes them asynchronously with an internal queueing system.
- **Amazon SageMaker Batch Transform** - Use if you do not need near-instantaneous response from the ML model and can gather data points together into a large batch for a schedule-based

inference. When a batch transform job starts, SageMaker initializes compute instances and distributes the inference or preprocessing workload among them. SageMaker Batch Transform automatically splits input files into mini-batches (so that you won't need to worry about out-of-memory (OOM) for large datasets) and shuts down compute instances once the entire dataset is processed.

Documents

- Amazon SageMaker Real-time Inference
- Amazon SageMaker Serverless Inference
- Amazon SageMaker Asynchronous Inference
- Amazon SageMaker Batch Transform

Blogs

- Announcing managed inference for Hugging Face models in Amazon SageMaker
- Performing batch inference with TensorFlow Serving in Amazon SageMaker
- Deploying ML models using SageMaker Serverless Inference
- Run computer vision inference on large videos with Amazon SageMaker asynchronous endpoints

Videos

- AWS re:Invent 2021 - Achieve high performance and cost-effective model deployment
- AWS re:Invent 2021 - Amazon SageMaker serverless inference

Examples

- Deploy models with Amazon SageMaker

Cost optimization pillar - Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable

you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment (ROI).

Best practices

- MLCOST-24: Use appropriate deployment option
- MLCOST-25: Explore cost effective hardware options
- MLCOST-26: Right-size the model hosting instance fleet

MLCOST-24: Use appropriate deployment option

Use real-time inference for low latency and ultra-high throughput for use cases with steady traffic patterns. Use batch transform for offline inference on data batches for use cases with large datasets. Deploy models at edge to optimize, secure, monitor, and maintain machine learning models on fleets of edge devices such as smart cameras, robots, personal computers, and mobile devices.

Implementation plan

- **Use Amazon SageMake**r - Amazon SageMaker has a broad selection of ML infrastructure and model deployment options to make it easy to deploy ML models at the best price-performance for any use case. It is a fully managed service and integrates with MLOps tools, so you can scale your model deployment, reduce inference costs, manage models more effectively in production, and reduce operational burden.
  - **Use Amazon SageMaker Real-time Inference, Amazon SageMaker Serverless Inference, Amazon SageMaker Asynchronous Inference, and Amazon SageMaker Batch Transform** - See “MLPER-11: Evaluate cloud versus edge options for machine learning deployment“.
  - **Use Amazon SageMaker Multi-Model endpoints** - Multi-model endpoints provide a scalable and cost-effective solution to deploying large numbers of models. They use a shared serving container that is enabled to host multiple models. This approach reduces hosting costs by improving endpoint utilization compared with using single-model endpoints. It also reduces deployment overhead because Amazon SageMaker manages loading models in memory and scaling them based on the traffic patterns to them.
  - **Use Amazon SageMaker multi-container endpoints** - SageMaker multi-container endpoints enable you to deploy multiple containers that use different models or frameworks on a single SageMaker endpoint. The containers can be run in a sequence as an inference pipeline, or

each container can be accessed individually by using direct invocation to improve endpoint utilization and optimize costs.

- **Use Amazon SageMaker Pipelines** - See **“MLREL-10: Automate endpoint changes through a pipeline“.**
- **Use Amazon SageMaker Edge** - See “Optimize model deployment on the edge” under “MLPER-10: Evaluate machine learning deployment option (cloud versus edge)”.

Documents

- Deploy models for inference
- SageMaker hosting options
- SageMaker Serverless Inference
- SageMaker Asynchronous Inference
- SageMaker Batch Transform
- Deploy models at the edge with SageMaker Edge Manager

Blogs

- Using Amazon SageMaker inference pipelines with multi-model endpoints
- Save on inference costs by using Amazon SageMaker multi-model endpoints
- Deploy multiple serving containers on a single instance using Amazon SageMaker multi-container endpoints
- Run computer vision inference on large videos with Amazon SageMaker asynchronous endpoints
- Batch Inference at Scale with Amazon SageMaker
- Amazon SageMaker Edge Manager Simplifies Operating Machine Learning Models on Edge Devices

Examples

- SageMaker Serverless Inference Walkthrough
- SageMaker Edge Manager Workshop

MLCOST-25: Explore cost effective hardware options

Machine learning models that power AI applications are becoming increasingly complex resulting in rising underlying compute infrastructure costs. Up to 90% of the infrastructure spend for developing and running ML applications is often on inference. Look for cost-effective infrastructure solutions for deploying their ML applications in production.

Implementation plan

- **Use Amazon SageMaker Neo** - Please see details of Amazon SageMaker Neo under "MLPER-10: Evaluate machine learning deployment option (cloud versus edge)". For inference in the cloud, SageMaker Neo speeds up inference and saves cost by creating an inference optimized container in SageMaker hosting. For inference at the edge, SageMaker Neo saves developers months of manual tuning by automatically tuning the model for the selected operating system and processor hardware.
- **Use Amazon SageMaker Elastic Inference** - Amazon Elastic Inference (EI) is a service that lets you attach just the right amount of GPU-powered inference acceleration to any EC2 instance. By using Amazon EI, you can speed up the throughput and decrease the latency of getting real-time inferences from your deep learning models that are deployed as Amazon SageMaker hosted models, but at a fraction of the cost of using a GPU instance for your endpoint. Add an Amazon EI accelerator in one of the available sizes to a deployable model in addition to a CPU instance type, and then add that model as a production variant to an endpoint configuration that you use to deploy a hosted endpoint. You can also add an Amazon EI accelerator to a SageMaker notebook instance so that you can test and evaluate inference performance when you are building your models.
- **Use Amazon EC2 Inf1 Instances** - Amazon EC2 Inf1 instances deliver high-performance ML inference at the lowest cost in the cloud. They deliver up to 2.3-times higher throughput and up to 70% lower cost per inference than comparable current generation GPU-based Amazon EC2 instances. Inf1 instances are built from the ground up to support machine learning inference applications. They feature up to 16 AWS Inferentia chips, high-performance machine learning inference chips designed and built by AWS. Additionally, Inf1 instances include second generation Intel Xeon Scalable processors and up to 100 Gbps networking to deliver high throughput inference.

Documents

- Use Amazon SageMaker Elastic Inference

- What Is Amazon Elastic Inference?
- Optimize model performance using Neo
- Amazon EC2 Inf1 Instances

Blogs

- Increasing performance and reducing the cost of MXNet inference using Amazon SageMaker Neo and Amazon Elastic Inference
- Reduce ML inference costs on Amazon SageMaker with hardware and software acceleration
- Announcing availability of Inf1 instances in Amazon SageMaker for high performance and cost-effective machine learning inference

Examples

- Increasing performance and reducing cost of deep learning inference using Amazon SageMaker Neo and Amazon Elastic Inference

MLCOST-26: Right-size the model hosting instance fleet

Use efficient compute resources to run models in production. In many cases, up to 90% of the infrastructure spend for developing and running an ML application is on inference, making it critical to use high-performance, cost-effective ML inference infrastructure. Selecting the right way to host and the right type of instance can have a large impact on the total cost of ML projects. Use automatic scaling (autoscaling) for your hosted models. *Auto scaling* dynamically adjusts the number of instances provisioned for a model in response to changes in your workload.

Implementation plan

- **Use Amazon SageMaker Inference Recommender** - Amazon SageMaker Inference Recommender automatically selects the right compute instance type, instance count, container parameters, and model optimizations for inference to maximize performance and minimize cost. You can use SageMaker Inference Recommender from SageMaker Studio, the AWS Command Line Interface (AWS CLI), or the AWS SDK, and within minutes, get recommendations to deploy your ML model. You can then deploy your model to one of the recommended instances or run a fully managed load test on a set of instance types you choose without worrying about testing infrastructure. You can review the results of the load test in SageMaker Studio and evaluate

the tradeoffs between latency, throughput, and cost to select the most optimal deployment configuration.

- **Use AutoScaling with Amazon SageMaker** - Amazon SageMaker supports an Autoscaling feature that monitors your workloads and dynamically adjusts the capacity to maintain steady and predictable performance at the lowest possible cost. When the workload increases, autoscaling brings more instances online. When the workload decreases, autoscaling removes unnecessary instances, helping you reduce your compute cost. SageMaker automatically attempts to distribute your instances across Availability Zones. So, we strongly recommend that you deploy multiple instances for each production endpoint for high availability. If you're using a VPC, configure at least two subnets in different Availability Zones so Amazon SageMaker can distribute your instances across those Availability Zones.

Documents

- Automatically Scale Amazon SageMaker Models
- Amazon SageMaker Inference Recommender

Blogs

- Ensure efficient compute resources on Amazon SageMaker
- Configuring autoscaling inference endpoints in Amazon SageMaker
- Announcing Amazon SageMaker Inference Recommender

Examples

- Right-sizing your Amazon SageMaker Endpoints
- Automatically Scale Amazon SageMaker Models
- SageMaker Inference Recommender

Sustainability pillar - Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage.

Best practices

- MLSUS-11: Align SLAs with sustainability goals
- MLSUS-12: Use efficient silicon
- MLSUS-13: Optimize models for inference
- MLSUS-14: Deploy multiple models behind a single endpoint

Related best practices

- **Allow automatic scaling of the model endpoint** (MLREL-11) - Configure automatic scaling for Amazon SageMaker Endpoints or use Serverless Inference and make efficient use of GPU with Amazon Elastic Inference. Elastic Inference allows you to attach just the right amount of GPU-powered inference acceleration to any EC2 or SageMaker instance type or ECS task. While training jobs process hundreds of data samples in parallel, inference jobs usually process a single input in real time, and thus consume a small amount of GPU compute. Amazon Elastic Inference allows you to reduce the cost and environmental impact of your inference by using GPU resources more efficiently.
- **Evaluate machine learning deployment option (cloud versus edge)** (MLPER-11) - When working on IoT use-cases, evaluate if running ML inference at the edge can reduce the environmental impact of your workload. For that, consider factors like the compute capacity of your devices, their energy consumption or the emissions related to data transfer to the Cloud. When deploying ML models to edge devices, consider using Amazon SageMaker Edge Manager which integrates with SageMaker Neo and AWS IoT GreenGrass.
- **Select optimal computing instance size** (MLCOST-09) - Amazon SageMaker Inference Recommender automates load testing and model tuning across SageMaker ML instances. It helps you select the best instance type and configuration (such as instance count, container parameters, and model optimizations) to ensure the maximum efficiency of the resources provisioned for inference.

MLSUS-11: Align SLAs with sustainability goals

Define service level agreements (SLAs) that support your sustainability goals while meeting your business requirements. Define SLAs to meet your business requirements, not exceed them. Make trade-offs that significantly reduce environmental impacts in exchange for acceptable decreases in service levels.

Implementation plan

- **Queue incoming requests and process them asynchronously** - If your users can tolerate some latency, deploy your model on serverless or asynchronous endpoints to reduce resources that are idle between tasks and minimize the impact of load spikes. These options will automatically scale the instance or endpoint count to zero when there are no requests to process, so you only maintain an inference infrastructure when your endpoint is processing requests.
- **Adjust availability** - If your users can tolerate some latency in the rare case of a failover, don't provision extra capacity. If an outage occurs or an instance fails, Amazon SageMaker automatically attempts to distribute your instances across Availability Zones. Adjusting availability is an example of a conscious trade off you can make to meet your sustainability targets.
- **Adjust response time** - When you don't need real-time inference, use SageMaker Batch Transform. Unlike a persistent endpoint, clusters are decommissioned when batch transform jobs finish so you don't continuously maintain an inference infrastructure.

Documents

- Amazon SageMaker Asynchronous inference
- Amazon SageMaker Batch Transform
- Optimize software and architecture for asynchronous and scheduled jobs
- Best practices for deploying models on SageMaker Hosting Services
- Align SLAs with sustainability goals
- Sustainability as a non-functional requirement

Blogs

- Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring

Videos

- AWS re:Invent 2021 - Architecting for sustainability - Optimize capacity for Sustainability

MLSUS-12: Use efficient silicon

Use the most efficient instance type compatible with your ML workload.

Implementation plan

AWS offers several purpose-built compute architectures that are optimized to minimize the sustainability impact of ML workloads:

- **For CPU-based ML inference, use AWS Graviton3** - These processors offer the best performance per watt in Amazon EC2. They use up to 60% less energy than comparable EC2 instances. Graviton3 processors deliver up to three times better performance compared to Graviton2 processors for ML workloads, and they support bfloat16.
- **For deep learning inference, use AWS Inferentia** - The Amazon EC2 Inf2 instances offer up to 50% better performance/watt over comparable Amazon EC2 instances because they and the underlying Inferentia2 accelerators are purpose built to run DL models at scale. Inf2 instances help you meet your sustainability goals when deploying ultra-large models.
- **For training, use AWS Trainium** - The Amazon EC2 trn1 instances based on the custom designed AWS Trainium chips offer up to 50% cost-to-train savings over comparable Amazon EC2 instances. When using a Trainium-based instance cluster, the total energy consumption for training BERT Large from scratch is approximately 25% lower compared to a same-sized cluster of comparable accelerated EC2 instances.

Documents

- Instances with AWS Inferentia
- Use instance types with the least impact

Blogs

- Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring
- Achieving 1.85x higher performance for deep learning based object detection with an AWS Neuron compiled YOLOv4 model on AWS Inferentia
- Deploying TensorFlow OpenPose on AWS Inferentia-based Inf1 instances for significant price performance improvements
- Amazon EC2 Update – Inf1 Instances with AWS Inferentia Chips for High Performance Cost-Effective Inferencing

MLSUS-13: Optimize models for inference

Improve efficiency of your models and thus use less resources for inference by compiling the models into optimized forms.

Implementation plan

- **Use open-source model compilers** - Libraries such as Treelite (for decision tree ensembles) improve the prediction throughput of models, due to more efficient use of compute resources.
- **Use third-party tools** - Solutions like Hugging Face Infinity allow you to accelerate transformer models and run inference not only on GPUs but also on CPUs.
- **Use Amazon SageMaker Neo** - SageMaker Neo enables developers to optimize ML models for inference on SageMaker in the cloud and supported devices at the edge. SageMaker Neo runtime consumes as little as one-tenth the footprint of a deep learning framework while optimizing models to perform up to 25 times faster with no loss in accuracy.

Documents

- Optimize model performance using Neo

Blogs

- Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring
- Unlock near 3x performance gains with XGBoost and Amazon SageMaker Neo

MLSUS-14: Deploy multiple models behind a single endpoint

Host multiple models behind a single endpoint to improve endpoint utilization. Sharing endpoint resources is more sustainable and less expensive than deploying a single model behind one endpoint.

Implementation plan

Amazon SageMaker provides three methods to deploy multiple models to a single endpoint:

- **Host multiple models in one container behind one endpoint** (MLCOST-24) - SageMaker multi-model endpoints (MME) are served using a single container. This feature is ideal when you have a large number of similar models that you can serve through a shared serving container and don't

need to access all the models at the same time. This can help cut inference costs and reduce carbon emissions by up to 90% .

- **Host multiple models which use different containers behind one endpoint** (MLCOST-24) – SageMaker multi-container endpoint (MCE) support deploying up to 15 containers that use different models or framework on a single endpoint, and invoking them independently or in sequence for low-latency inference and cost savings. The models can be completely heterogenous, with their own independent serving stack.
- **Use SageMaker inference pipelines** - An *inference pipeline* is an Amazon SageMaker model that is composed of a linear sequence of containers deployed behind a single endpoint. You can use an inference pipeline to combine preprocessing, predictions, and post-processing data science tasks. The output from the one container is passed as input to the next. When defining the containers for a pipeline model, you also specify the order in which the containers are run.

Documents

- Host multiple models in one container behind one endpoint
- Host multiple models which use different containers behind one endpoint
- Host models along with pre-processing logic as serial inference pipeline behind one endpoint

Blogs

- Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring
- Save on inference costs by using Amazon SageMaker multi-model endpoints

ML lifecycle phase – Monitoring

The model monitoring system must capture data, compare that data to the training set, define rules to detect issues, and send alerts. This process repeats on a defined schedule, when initiated by an event, or when initiated by human intervention. The issues detected in the monitoring phase include: data quality, model quality, bias drift, and feature attribution drift.

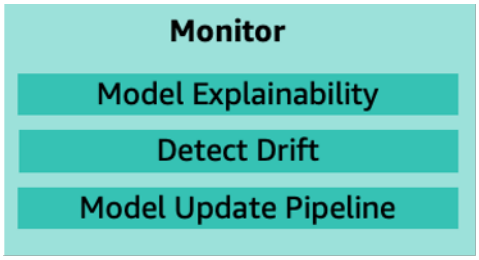

Figure 18: Post deployment monitoring - main components

Figure 18 lists key components of monitoring, including:

- **Model explainability** - Monitoring system uses *explainability* to evaluate the soundness of the model and if the predictions can be trusted.
- **Detect drift** - Monitoring system detects data and concept drifts, initiates an alert, and sends it to the alarm manager system. Data drift is significant changes to the data distribution compared to the data used for training. Concept drift is when the properties of the target variables change. Any kind of drift results in model performance degradation.
- **Model update pipeline** - If the alarm manager identifies any violations, it launches the model update pipeline for a re-train. This can be seen in Figure 19. The *Data prepare*, *CI/CD/CT*, and *Feature* pipelines will also be active during this process.

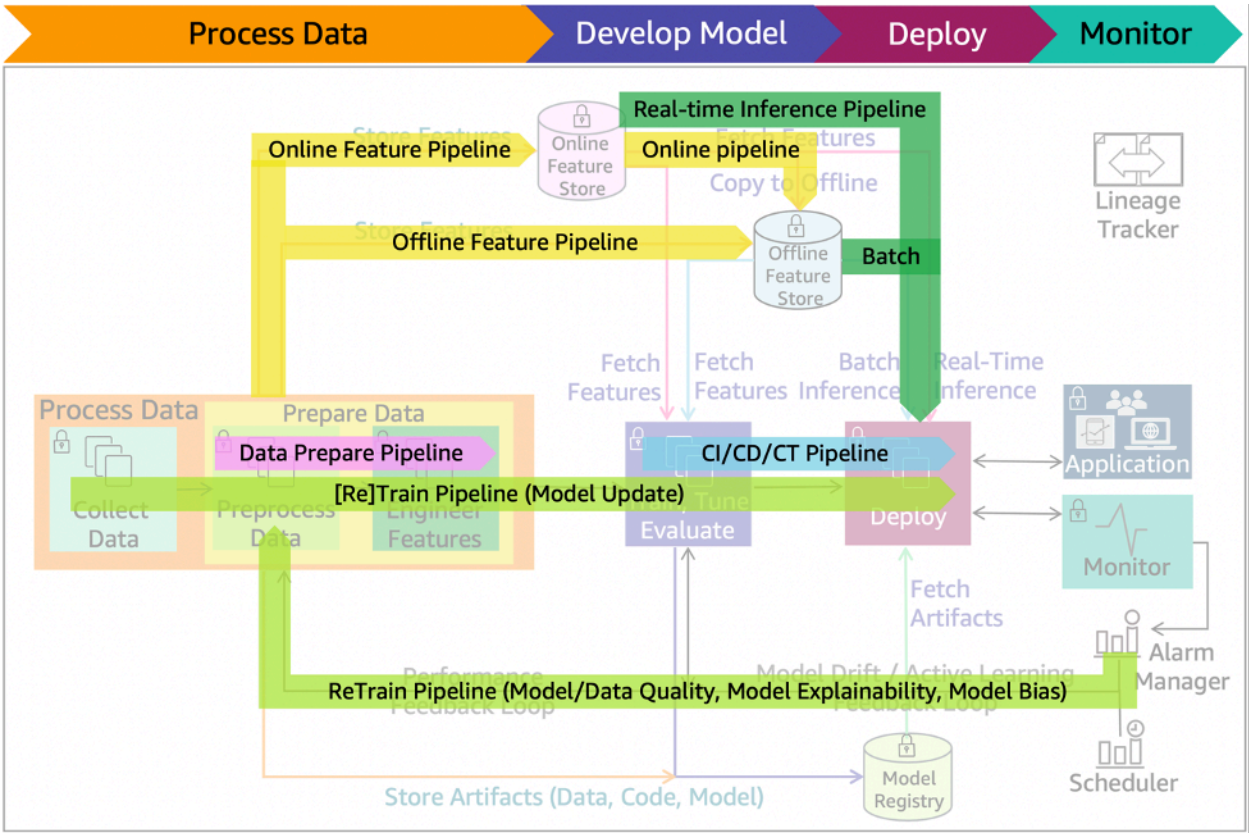

Figure 19: ML lifecycle with model update re-train and batch/real-time inference pipelines

Best practices

- Operational excellence pillar – Best practices
- Security pillar – Best practices
- Reliability pillar – Best practices
- Performance efficiency pillar – Best practices
- Cost optimization pillar – Best practices
- Sustainability pillar – Best practices

Operational excellence pillar – Best practices

The operational excellence pillar includes the ability to run and monitor systems to deliver business value and to continually improve supporting processes and procedures. This section includes best practices to consider for monitoring models in production.

Best practices

- MLOE-15: Enable model observability and tracking
- MLOE-16: Synchronize architecture and configuration, and check for skew across environments

MLOE-15: Enable model observability and tracking

Establish model monitoring mechanisms to identify and proactively avoid any inference issues. ML models can degrade in performance over time due to drifts. Monitor metrics that are attributed to your model's performance. For real time inference endpoints, measure the operational health of the underlying compute resources hosting the endpoint and the health of endpoint responses. Establish lineage to trace hosted models back to versioned inputs and model artifacts for analysis.

Implementation plan

- **Use Amazon SageMaker Model Monitor** - Continually monitor the quality of Amazon SageMaker ML models in production and compare with the results from training using SageMaker Model Monitor.
- **Use Amazon CloudWatch** - Amazon SageMaker Model Monitor automatically sends metrics to Amazon CloudWatch so that you can gather and analyze usage statistics for your ML models.
- **Use SageMaker Model Dashboard** - View, search, and explore your models in a centralized portal from the SageMaker console. Set up monitors with Amazon SageMaker Model Monitor and track the performance of your models that are hosted on real-time inference endpoints. Find models that violate thresholds you have set for data quality, model quality, bias, and explainability
- **Use Amazon SageMaker Clarify** - Identify various types of bias that can emerge during model training or when the model is in production. This helps improve your ML models by detecting potential bias and helping explain the predictions that the models make. SageMaker Clarify helps explain how these models make predictions using a feature attribution approach. It also monitors inferences that the models make in production for bias drift or feature attribution drift. SageMaker Clarify provides tools to help you generate model governance reports that you can use to inform risk and compliance teams, and external regulators.
- **Track your model pipeline with SageMaker ML lineage Tracking** – Lineage tracking creates and stores information about the steps of a machine learning workflow from data preparation to model deployment. Keep a running history of model discovery experiments. Establish model governance by tracking model lineage artifacts for auditing and compliance verification.
- **Use SageMaker Model Cards to simplify model information gathering** – Documentation on model information, such as business requirements, key decisions, and observations during model

development and evaluation, is required to support approval workflows, registration, audits, customer inquiries, and monitoring. Amazon SageMaker Model Cards provide a single location to store model information (for example, performance goals, and risk rating) and training and evaluation results (for example, bias or accuracy measurements) in the AWS Management Console, streamlining documentation throughout a model's lifecycle.

- **Use the automated validation capability of Amazon SageMaker** – Amazon SageMaker Inference enables you to compare the performance of new models against production models, using the same real-world inference request data in real time. Amazon SageMaker can be used to route a copy of the inference requests received by the production model to the new model and generate a dashboard to display performance differences across key metrics in real time.

Documents

- Amazon SageMaker Model Monitor
- Monitoring Amazon ML with Amazon CloudWatch Metrics
- How Amazon CloudWatch works
- Clarify, Fairness and Explainability
- SageMaker Model Dashboard
- Amazon SageMaker Model Cards
- Amazon SageMaker Shadow Testing

Blogs

- Architect and build the full machine learning lifecycle with AWS: An end-to-end Amazon SageMaker demo
- Improve governance of your machine learning models with Amazon SageMaker
- New for Amazon SageMaker – Perform Shadow Tests to Compare Inference Performance Between ML Model Variants

Videos

- Introducing Amazon SageMaker Clarify, part 1 - Bias detection- AWS re:Invent 2020
- Introducing Amazon SageMaker Clarify, part 2 - Model explainability - AWS re:Invent 2020
- AWS re:Invent 2020: Understand ML model predictions & biases with Amazon SageMaker Clarify

- AWS re:Invent 2022 - Minimizing the production impact of ML model updates with shadow testing

Examples

- Monitoring bias drift and feature attribution drift with Amazon SageMaker Clarify

MLOE-16: Synchronize architecture and configuration, and check for skew across environments

Ensure that all systems and configurations are identical across development and deployment phases. Otherwise, the same algorithm can result in different inference results depending on differences in system architectures. Ensure that the model gets the same range of accuracy in development, staging, and production environments. Perform this check as part of the normal promotion process.

Implementation plan

- **Use AWS CloudFormation** - AWS CloudFormation gives you an easy way to model a collection of related AWS and third-party resources. CloudFormation provisions these resources quickly and consistently, and manages them throughout their lifecycles by treating infrastructure as code. You can use a CloudFormation template to create, update, and delete an entire stack as a single unit, as often as you need to, instead of managing resources individually. You also can manage and provision stacks across multiple AWS accounts and AWS Regions. This will synchronize your architecture and configuration across environments.
- **Use Amazon SageMaker Model Monitor** -Continually monitor the quality of Amazon SageMaker machine learning models in production and compare with the results from training using SageMaker Model Monitor. Set alerts that notify you when there are deviations in the model quality. Early and proactive detection of these deviations enables you to take corrective actions. These actions can include retraining models, auditing upstream systems, and fixing quality issues without having to monitor models manually or build additional tools. Model Monitor provides monitoring capabilities that do not require coding; you also have the flexibility to monitor models by adding code to provide custom analysis.

Documents

- Amazon SageMaker Model Monitor

- Implement infrastructure as code

Blogs

- Amazon SageMaker Model Monitor – Fully Managed Automatic Monitoring for your Machine Learning Models
- AWS CloudFormation – Create Your AWS Stack From a Recipe

Examples

- Introduction to Amazon SageMaker Model Monitor
- Model Monitor Visualization

Security pillar – Best practices

The security pillar encompasses the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. This section includes best practices to consider when monitoring models in production.

Best practices

- MLSEC-12: Restrict access to intended legitimate consumers
- MLSEC-13: Monitor human interactions with data for anomalous activity

MLSEC-12: Restrict access to intended legitimate consumers

Use least-privileged permissions to invoke the deployed model endpoint. For consumers who are external to the workload environment, provide access via a secure API.

Implementation plan

- **Use secure inference API endpoints** - Host the model so that a consumer of the model can perform inference against it securely. Enable consumers using the API to define the relationship, restrict access to the base model, and provide monitoring of model interactions.
- **Secure inference endpoints** - Only authorized parties should make inferences against the ML model. Treat inference endpoints as you would any other HTTPS API. Ensure that you follow guidance from the AWS Well-Architected Framework to provide network controls, such as

restricting access to specific IP ranges, and bot control. The HTTPS requests for these API calls should be signed, so that the requester identity can be verified, and the requested data is protected in transit.

Documents

- Amazon SageMaker: Real-time Inference
- Give SageMaker Hosted Endpoints Access to Resources in Your Amazon VPC
- Register and Deploy Models with Model Registry

Blogs

- Integrating machine learning models into your Java-based microservices
- How Financial Institutions can use AWS to Address Regulatory Reporting
- Secure deployment of Amazon SageMaker resources

Videos

- AWS re:Invent 2019: End-to-End machine learning using Spark and Amazon SageMaker

Examples

- Amazon SageMaker secure MLOps
- Accelerating Machine Learning Development with Data Science as a Service from Change Healthcare

MLSEC-13: Monitor human interactions with data for anomalous activity

Ensure that data access logging is enabled. Audit for anomalous data access events, such as access events from abnormal locations, or activity exceeding the baseline for that entity. Use services and tools that support anomalous activity alerting, and combine their use with data classification to assess risk. Evaluate using services to aid in monitoring data access events.

Implementation plan

- **Enable data access logging** - Verify that you have data access logging for all human CRUD (create, read, update, and delete) operations, including the details of who accessed what elements, what action they took, and at what time.
- **Classify your data** - Use Amazon Macie for protecting and classifying training and inference data in Amazon S3. Amazon Macie is a fully managed security service. It uses ML to automatically discover, classify, and protect sensitive data in AWS. The service recognizes sensitive data, such as personally identifiable information (PII) or intellectual property.
- **Monitor and protect** - Use Amazon GuardDuty to monitor for malicious and unauthorized activities. This will enable protecting AWS accounts, workloads, and data stored in Amazon S3.

Documents

- Amazon SageMaker Incident Response - Logging & Monitoring
- Amazon GuardDuty S3 Finding Types - which aid in anomaly detection for S3 resource access events.

Blogs

- Building a Self-Service, Secure, & Continually Compliant Environment on AWS
- How to Use New Advanced Security Features for Amazon Cognito user pools
- Best practices for setting up Amazon Macie with AWS Organizations

Videos

- Protect Your Data in S3 with Amazon Macie and Amazon GuardDuty - AWS Online Tech Talks
- AWS re:Invent 2020: Protecting sensitive data with Amazon Macie and Amazon GuardDuty

Examples

- Controlling and auditing data exploration activities with Amazon SageMaker Studio and AWS Lake Formation

Reliability pillar – Best practices

The reliability pillar encompasses the ability of a workload to perform its intended function correctly and consistently when it's expected to. This section includes best practices to consider while monitoring deployed models in production.

Best practices

- MLREL-12: Allow automatic scaling of the model endpoint
- MLREL-13: Ensure a recoverable endpoint with a managed version control strategy

MLREL-12: Allow automatic scaling of the model endpoint

Implement capabilities that allow the automatic scaling of model endpoints. This helps ensure the reliable processing of predictions to meet changing workload demands. Include monitoring on endpoints to identify a threshold that initiates the addition or removal of resources to support current demand.

After a request to scale is received, put in place a solution to scale backend resources supporting that endpoint.

Implementation plan

- **Configure automatic scaling for Amazon SageMaker Endpoints**- Amazon SageMaker supports automatic scaling (autoscaling) for your hosted models. SageMaker Endpoints can be configured with autoscaling. This ensures that as traffic increases in your application your endpoint can maintain the same level of service availability. Automatic scaling is a key feature of the cloud. It allows you to automatically provision new resources horizontally to handle increased user demand or system load. Automatic scaling is also a key component of creating event-driven architectures and is a necessary capability of any distributed system.
- **Use Amazon Elastic Inference**- With Amazon Elastic Inference, you can choose the CPU instance in AWS that is best suited to the overall compute and memory needs of your application. Separately configure the right amount of GPU-powered inference acceleration, allowing you to efficiently utilize resources and reduce costs.
- **Use Amazon Elastic Inference with EC2 Auto Scaling** - When you create an Auto Scaling group, you can specify the information required to configure the Amazon EC2 instances. This includes Elastic Inference accelerators. To do this, specify a launch template with your instance configuration and the Elastic Inference accelerator.

Documents

- Automatically Scale Amazon SageMaker Model
- What is Amazon Elastic Inference?

Blogs

- Configuring autoscaling inference endpoints in Amazon SageMaker

Videos

- Build, Train and Deploy ML Models at Scale with Amazon SageMaker
- Deploy Your ML Models to Production at Scale with Amazon SageMaker

Examples

- Automatically Scale Amazon SageMaker Models

MLREL-13: Ensure a recoverable endpoint with a managed version control strategy

Ensure an endpoint responsible for hosting model predictions, and all components responsible for generating that endpoint, are fully recoverable. Some of these components include model artifacts, container images, and endpoint configurations. Ensure all required components are version controlled, and traceable in a lineage tracker system.

Implementation plan

- **Implement MLOps best practices with Amazon SageMaker Pipelines and Projects** - Amazon SageMaker Pipelines is a service for building machine learning pipelines. It automates developing, training, and deploying models in a versioned, predictable manner. Amazon SageMaker Projects enable teams of data scientists and developers to collaborate on machine learning business problems. A SageMaker project is an Service Catalog provisioned product that enables you to easily create an end-to-end ML solution. SageMaker Projects entities include pipeline executions, registered models, endpoints, datasets, and code repositories.

- **Use infrastructure as code (IaC) tools** - Use AWS CloudFormation to define and build your infrastructure, including your model endpoints. Store your CloudFormation code in git repositories, such as AWS CodeCommit, so that you can version control your infrastructure code.
- **Use Amazon Elastic Container Registry (Amazon ECR)** - Store your containers in Amazon ECR, an artifact repository for Docker containers. Amazon ECR automatically creates a version hash for your containers as you update them, allowing you to roll back to previous versions.

Documents

- AWS CloudFormation
- Infrastructure as Code
- SageMaker Pipelines Overview
- What is AWS CodeCommit?
- What is a SageMaker Project?
- What is Service Catalog?
- What is Amazon Elastic Container Registry

Blogs

- How to manage Amazon SageMaker code with AWS CodeCommit
- Building, automating, managing, and scaling ML workflows using Amazon SageMaker Pipelines
- Multi-account model deployment with Amazon SageMaker Pipelines
- Automate feature engineering pipelines with Amazon SageMaker

Videos

- Infrastructure as Code on AWS - AWS Online Tech Talks
- AWS re:Invent 2020: How to create fully automated ML workflows with Amazon SageMaker Pipelines
- Introducing Amazon SageMaker Pipelines - AWS re:Invent 2020
- AWS re:Invent 2020: Implementing MLOps practices with Amazon SageMaker

Examples

- CI/CD Pipeline for AWS CloudFormation templates on AWS
- Amazon SageMaker MLOps

Performance efficiency pillar – Best practices

The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and the maintenance of that efficiency as demand changes and technologies evolve. This section includes best practices to consider while monitoring models in production.

Best practices

- MLPER-13: Evaluate model explainability
- MLPER-14: Evaluate data drift
- MLPER-15: Monitor, detect, and handle model performance degradation
- MLPER-16: Establish an automated re-training framework
- MLPER-17: Review for updated data/features for retraining
- MLPER-18: Include human-in-the-loop monitoring

MLPER-13: Evaluate model explainability

Evaluate model performance as constrained by the explainability requirements of the business. Compliance requirements, business objectives, or both might require that the inferences from a model be directly explainable. Evaluate the explainability needs, and the trade-off between explainability and model complexity. Then select the model type or evaluation metrics. This approach provides transparency into the reasons that a particular inference was attained given the input data.

Implementation plan

- **Use Amazon SageMaker Clarify to explain model results** - Amazon SageMaker Clarify helps improve your ML models by detecting potential bias and helping explain the predictions that models make. It helps you identify various types of bias in data that can emerge during model training or in production. SageMaker Clarify helps explain how these models make predictions using a feature attribution approach. It also monitors inferences that the models make in production for bias or feature attribution drift. The fairness and explainability functions provided

by SageMaker Clarify help you build less biased and more understandable machine learning models. It also provides tools to help you generate model governance reports that you can use to inform risk and compliance teams, and external regulators.

Documents

- Amazon SageMaker Clarify Model Explainability
- Feature Attributions that Use Shapley Values
- What is Fairness and Model Explainability for Machine Learning Predictions?

Blogs

- ML model explainability with Amazon SageMakerClarify and the SKLearn pre-built container
- Explaining Amazon SageMakerAutopilot models with SHAP
- Human-in-the-loop review of model explanations with Amazon SageMakerClarify and Amazon A2I

Videos

- Interpretability and explainability in machine learning
- Explaining Credit Decisions with Amazon SageMaker

Examples

- Fairness and Explainability with SageMaker Clarify
- Explainability with Amazon SageMaker Debugger

MLPER-14: Evaluate data drift

Understand the effects of data drift on model performance. In cases where the data has drifted, the model could generate inaccurate predictions. Consider a strategy that monitors and adapts to data drift through re-training.

Implementation plan

- **Use Amazon SageMaker Model Monitor, and SageMaker Clarify**- Amazon SageMaker Model Monitor helps you maintain high-quality ML models by detecting model and concept drift in real time, and sending you alerts so you can take immediate action. Model and concept drift are detected by monitoring the quality of the model. Independent variables (also known as features) are the inputs to an ML model, and dependent variables are the outputs of the model. Additionally, SageMaker Model Monitor is integrated with Amazon SageMaker Clarify to help you identify potential bias in your ML models.

Documents

- Amazon SageMaker Model Monitor
- Amazon SageMaker Clarify - Model Explainability

Blogs

- Amazon SageMaker Clarify Detects Bias and Increases the Transparency of Machine Learning Models

Videos

- Detect machine learning (ML) model drift in production

Examples

- Amazon SageMaker Clarify

MLPER-15: Monitor, detect, and handle model performance degradation

Model performance could degrade over time for reasons such as data quality, model quality, model bias, and model explainability. Continuously monitor the quality of the ML model in real time. Identify the right time and frequency to retrain and update the model. Configure alerts to notify and initiate actions if any drift in model performance is observed.

Implementation plan

- **Monitor model performance** - Amazon SageMaker Model Monitor continually monitors the quality of Amazon SageMaker machine learning models in production. Establish a baseline during training before model is in production. Collect data while in production and compare changes in model inferences. Observations of drifts in the data statistics will indicate that the model may need to be retrained. The timing of drifts will establish a schedule for retraining. Use SageMaker Clarify to identify model bias. Configure alerting systems with Amazon CloudWatch to send notifications for unexpected bias or changes in data quality.
- **Perform automatic scaling** - Amazon SageMaker includes automatic scaling capabilities for your hosted model to dynamically adjust underlying compute supporting an endpoint based on demand. This capability ensures that your endpoint can dynamically support demand while reducing operational overhead.
- **Monitor endpoint metrics** - Amazon SageMaker also outputs endpoint metrics for monitoring the usage and health of the endpoint. Amazon SageMaker Model Monitor provides the capability to monitor your ML models in production and provides alerts when data quality issues appear. Create a mechanism to aggregate and analyze model prediction endpoint metrics using services, such as Amazon OpenSearch Service (OpenSearch Service). OpenSearch Service supports Kibana for dashboards and visualization. The traceability of hosting metrics back to versioned inputs allows for analysis of changes that could be impacting current operational performance.

Documents

- Amazon SageMaker Model Monitor
- How Amazon CloudWatch works
- Fairness and Explainability with SageMaker Clarify

Blogs

- Monitoring in-production ML models at large scale using Amazon SageMaker Model Monitor
- ML model explainability with Amazon SageMaker Clarify and the SKLearn pre-built container

Videos

- Understand ML model predictions & biases with Amazon SageMaker Clarify
- Deep Dive on Amazon SageMaker Debugger & Amazon SageMaker Model Monitor

- Detect machine learning (ML) model drift in production

Examples

- Amazon SageMaker Model Monitor Examples- Github

MLPER-16: Establish an automated re-training framework

Monitor the data and the model predictions. Run analyses of model performance against defined metrics to identify errors due to data and concept drift. Automate model re-training to mitigate these errors on fixed scheduled intervals, or when model variance reaches a defined threshold. Automated model retraining can also be started as enough new data becomes available.

Implementation plan

- **Identify retraining opportunities** - Monitor data statistics and ML inferences at production using Amazon SageMaker Model Monitor. If the data drifts beyond a defined threshold, then start retraining. Additionally, retraining can be initiated at defined scheduled intervals (to meet business requirements) or when additional training data is available. AWS supports mechanisms for automatically starting retraining based on a new data *PUT* to an Amazon S3 bucket. Ensure model versioning is supported when incorporating additional data into your models. This enables re-creating an inadvertently deleted model artifact using the combined versions of components used to create the versioned artifact.
- **Use Amazon SageMaker Pipelines** - A retraining pipeline can be developed using Amazon SageMaker Pipelines that enables orchestration using step creation and management.
- **Use AWS Step Functions**- You can also use AWS Step Functions Data Science SDK for Amazon SageMaker to automate training of a machine learning model. Define all the steps in the workflow and set up alerts to start the flow. To detect the presence of new training data in an S3 bucket, AWS CloudTrail combined with Amazon CloudWatch Events allows you to start an AWS Step Function workflow to initiate retraining tasks in your training pipeline.
- **Use third-party tools** - Use third-party deployment orchestration tools, such as Jenkins, that integrate with AWS service APIs to automate model retraining when new data is available.

Documents

- Amazon SageMaker Model Building Pipelines

- Retraining Models on New Data
- Amazon SageMaker Model Monitor
- Train a Machine Learning Model (using AWS Step Functions)
- AWS Step Functions Data Science SDK for Python

Blogs

- Monitoring in-production ML models at large scale using Amazon SageMaker Model Monitor
- Automating model retraining and deployment using the AWS Step Functions Data Science SDK for Amazon SageMaker
- Automating complex deep learning model training using Amazon SageMaker Debugger and AWS Step Functions
- Build a CI/CD pipeline for deploying custom machine learning models using AWS services
- Create SageMaker Pipelines for training, consuming and monitoring your batch use cases

Videos

- How to create fully automated ML workflows with Amazon SageMaker Pipelines (29:23)
- Machine Learning and Automated Model Retraining with SageMaker

Examples

- Autopilot, Debugger and Model Monitor – Immersion Day
- Amazon SageMaker MLOps

MLPER-17: Review for updated data/features for retraining

Establish a framework to run data exploration and feature engineering at pre-determined time intervals based on data volatility and availability. New features that have not been considered in the model training can affect the accuracy of model inferences.

Implementation plan

- **Explore changing data with Amazon SageMaker Data Wrangler** - Evaluate the rate of change of the business environment to set a schedule for validating and possibly changing model input

data and features. Analyze the data using Amazon SageMaker Data Wrangler and explore new features. Establish a team who will periodically evaluate and possibly change features and retrain the model.

Documents

- Prepare ML Data with Amazon SageMaker Data Wrangler
- Amazon SageMaker Model Monitor

Blogs

- Exploratory data analysis, feature engineering, and operationalizing your data flow into your ML pipeline with Amazon SageMaker Data Wrangler

Videos

- Introducing Amazon SageMaker Data Wrangler – AWS re:Invent 2020

MLPER-18: Include human-in-the-loop monitoring

Use human-in-the-loop monitoring to monitor model performance efficiently. When automating decision processes, the human labeling of model results is a reliable quality test for model inferences.

Compare human labels with model inferences to estimate model performance degradation. Perform mitigation as model re-training.

Implementation plan

- **Use Amazon Augmented AI to get human review** - Learn how to design a quality assurance system for model inferences. Establish a team of subject matter experts to audit model inference in production. Use Amazon Augmented AI (Amazon A2I) to get human review of low-confidence predictions or random prediction samples. Amazon A2I uses resources in IAM, SageMaker, and Amazon S3 to create and run your human review workflows.

Documents

- Using Amazon Augmented AI for Human Review

Blogs

- Human-in-the-loop review of model explanations with Amazon SageMaker Clarify and Amazon A2I
- Verifying and adjusting your data labels to create higher quality training datasets with Amazon SageMaker Ground Truth

Videos

- Easily Implement Human in the Loop into Your Machine Learning Predictions with Amazon A2I

Cost optimization pillar – Best practices

The cost optimization pillar includes the continual process of refinement and improvement of a system over its entire lifecycle. From the initial design of your very first proof of concept to the ongoing operation of production workloads, adopting the practices in this document can enable you to build and operate cost-aware systems that achieve business outcomes and minimize costs, thus allowing your business to maximize its return on investment. This section includes best practices to consider for monitoring models in production.

Best practices:

- MLCOST-27: Monitor usage and cost by ML activity
- MLCOST-28: Monitor Return on Investment for ML models
- MLCOST-29: Monitor endpoint usage and right-size the instance fleet

MLCOST-27: Monitor usage and cost by ML activity

Use cloud resource tagging to manage, identify, organize, search for, and filter resources. Tags help categorize resources by purpose, owner, environment, or other criteria. Associate costs with resources using ML activity categories, such as re-training and hosting, by using tagging to manage and optimize cost in deployment phases. Tagging can be useful for generating billing reports with breakdown of cost by associated resources.

Implementation plan

- **Use AWS tagging** -A tag is a label that you or AWS assigns to an AWS resource. Each tag consists of a key and a value. For each resource, each tag key must be unique, and each tag key can have

only one value. You can use tags to organize your resources, and cost allocation tags to track your AWS costs on a detailed level. AWS uses the cost allocation tags to organize your resource costs on your cost allocation report. This will make it easier for you to categorize and track your AWS costs. AWS provides two types of cost allocation tags, an AWS-generated tag and user-defined tags.

- **Use AWS Budgets to keep track of cost** - AWS Budgets helps you track your Amazon SageMaker cost, including development, training, and hosting. You can also set alerts and get a notification when your cost or usage exceeds (or is forecasted to exceed) your budgeted amount. After you create your budget, you can track the progress on the AWS Budgets console.

Documents

- Tagging AWS resources
- Use Tags to Track and Allocate Amazon SageMaker Studio Notebooks Costs
- Tagging best practices
- Managing your costs with AWS Budgets

Blogs

- Optimizing costs for machine learning with Amazon SageMaker

Videos

- How can I tag my AWS resources to divide up my bill by cost center or project?

MLCOST-28: Monitor Return on Investment for ML models

Once a model is deployed into production, establish a reporting capability to track the value which is being delivered. For example:

- If a model is used to support customer acquisition: How many new customers are acquired and what is their spend when the model's advice is used compared with a baseline.
- If a model is used to predict when maintenance is needed: What savings are being made by optimizing the maintenance cycle.

Effective reporting enables you to compare the value delivered by an ML model against the ongoing runtime cost and to take appropriate action. If the ROI is substantially positive, are there ways in which this might be scaled to similar challenges, for example. If the ROI is negative, could this be addressed by remedial action, such as reducing the model latency by using server less inference, or reducing the run time cost by changing the compromise between model accuracy and model complexity, or layering in an additional simpler model to triage or filter the cases that are submitted to the full model.

Implementation plan

- **Use dashboard reporting** - Using a reporting tool such as Amazon QuickSight to develop business focused reports showing the value delivered by using the model in terms of business KPIs.

Documents

- Amazon QuickSight

MLCOST-29: Monitor endpoint usage and right-size the instance fleet

Ensure efficient compute resources are used to run models in production. Monitor your endpoint usage and right-size the instance fleet. Use automatic scaling (autoscaling) for your hosted models. *Autoscaling* dynamically adjusts the number of instances provisioned for a model in response to changes in your workload.

Implementation plan

- **Monitor Amazon SageMaker endpoints with Amazon CloudWatch** - You can monitor Amazon SageMaker using Amazon CloudWatch, which collects raw data and processes it into readable, near real-time metrics. Use metrics such as CPUUtilization, GPUUtilization, MemoryUtilization, GPUUtilization to view your endpoint's resource utilization and use the information to right-size the endpoint instance.
- **Use autoscaling with Amazon SageMaker** - Amazon SageMaker supports autoscaling that monitors your workloads and dynamically adjusts the capacity to maintain steady and predictable performance at the lowest possible cost. When the workload increases, autoscaling brings more instances online. When the workload decreases, autoscaling removes unnecessary instances, helping you reduce your compute cost. SageMaker automatically attempts to distribute your instances across Availability Zones. So, we strongly recommend that you

deploy multiple instances for each production endpoint for high availability. If you're using a VPC, configure at least two subnets in different Availability Zones so Amazon SageMaker can distribute your instances across those Availability Zones.

- **Determine the resource placement carefully** – Amazon FSx for Lustre can be an input data source for Amazon SageMaker. When FSx for Lustre is used as an input data source, Amazon SageMaker ML training jobs are accelerated by eliminating the initial Amazon S3 download step. However, as a best practice, it is recommended that customers deploy FSx for Lustre and SageMaker in the same Availability Zone. Deploying them across Availability Zones or VPC can result in a significant cost.

Documents

- Automatically Scale Amazon SageMaker Model
- Monitor Amazon SageMaker endpoints with Amazon CloudWatch

Blogs

- Use Amazon CloudWatch custom metrics for real-time monitoring of Amazon SageMaker model performance
- Speed up training on Amazon SageMaker using Amazon FSx for Lustre and Amazon EFS file systems

Sustainability pillar – Best practices

The sustainability pillar focuses on environmental impacts, especially energy consumption and efficiency, since they are important levers for architects to inform direct action to reduce resource usage. This section includes best practices to consider for monitoring models in production.

Best practices:

- MLSUS-15: Measure material efficiency
- MLSUS-16: Retrain only when necessary

MLSUS-15: Measure material efficiency

Measure efficiency of your workload in provisioned resources per unit of work, to measure not only the business success of the workload, but also its material efficiency. Use this measure as a baseline for your sustainability improvement process.

Implementation plan

- **Use provisioned resources per unit of work as a key performance indicator** - Divide the provisioned resources by the business outcomes achieved to determine the provisioned resources per unit of work. This allows you to normalize your sustainability KPIs and compare the performance over time.
- **Establish a baseline** - Understand the resources provisioned by your workload to complete a unit of work (for example, training the model or making a prediction)
- **Estimate improvement** - Estimate improvement as both the quantitative reduction in resources provisioned and the percentage change from your baseline resources provisioned per unit of work. Quantify the net benefit from your improvement over time to show the return on investment from your improvement activities.

Resources

- Measure results and replicate successes

Videos

- AWS re:Invent 2021 - Architecting for sustainability - Resources per Unit of Work

MLSUS-16: Retrain only when necessary

Because of model drift, robustness requirements, or new ground truth data being available, models usually need to be retrained. Instead of retraining arbitrarily, monitor your ML model in production, automate your model drift detection and only retrain when your model's predictive performance has fallen below defined KPIs.

Implementation plan

- **Determine key performance indicators** - With business stakeholders, identify a minimum acceptable accuracy and a maximum acceptable error.

- **Monitor your model deployed in Production** - Automate your model drift detection using Amazon SageMaker Model Monitor
- **Automate your retraining pipelines** - Use Amazon SageMaker Pipelines, AWS Step Functions Data Science SDK for Amazon SageMaker or third-party tools to automate your retraining pipelines.

Resources

- MLPER-01: Determine key performance indicators

Blogs

- Optimize AI/ML workloads for sustainability: Part 3, deployment and monitoring
- Monitoring in-production ML models at large scale using Amazon SageMaker Model Monitor
- Automating model retraining and deployment using the AWS Step Functions Data Science SDK for Amazon SageMaker
- Automate model retraining with Amazon SageMaker Pipelines when drift is detected

Conclusion

The Well-Architected ML design principles in this paper provide the guidance for the best practices collection. The technology and cloud agnostic best practices across the Well-Architected pillars provide architectural guidance for each phase of the ML lifecycle. Implementation plans provide guidance on implementing these best practices on AWS.

Architecture diagrams demonstrate the lifecycle phases with the supporting technologies, that enable many of the best practices introduced in this paper. The ML lens extends the Well-Architected Framework, and builds specific machine learning best practices upon it. As you work towards building and deploying production ML workloads in AWS, we recommend reviewing the AWS Well-Architected Framework pillar best practices.

Use the Lens to help ensure that your ML workloads are architected with operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability in mind. Plan early and make informed decisions when designing new workloads. Use the best practices to guide you through building and deploying new workloads faster. Using the lens guidance, evaluate existing workloads regularly to identify, mitigate, and address potential issues early.

Contributors

The following individuals and organizations contributed to this document:

- Benoit de Chateauvieux - Startup Solutions Architect, Amazon Web Services
- Dan Ferguson - Sr. Specialist Solutions Architect – PE, Amazon Web Services
- Michael Hsieh – Principal AI/ML Specialist SA, WWSO BDSI, Amazon Web Services
- Ganapathi Krishnamoorthi – Principal Startup Solutions Architect - AI/ML, Amazon Web Services
- Neil Mackin – Principal ML Strategist, AWS Customer Engineering, Amazon Web Services
- Haleh Najafzadeh - Senior Manager, Well-Architected Content, Amazon Web Services
- Raj Pathak – Senior Solutions Architect, WWCS Geo Solutions Architecture, Amazon Web Services
- Ram Pathangi – Solutions Architect, WWCS Geo Solutions Architecture, Amazon Web Services
- Raju Patil – Data Scientist, AWS WWCO Professional Services, Amazon Web Services
- Eddie Pick - Manager, Startup Solutions Architect, Amazon Web Services
- Deepali Rajale – Specialist Technical Account Manager – AI/ML (Sp), AWS Enterprise Support, Amazon Web Services
- Brendan Sisson - Principal Solutions Architect, Amazon Web Services
- Dhiraj Thakur – Senior Solutions Architect, Amazon Web Services
- Pallavi Nargund - Principal Solutions Architect, Amazon Web Services
- Patrick Roberts - Senior Data Scientist, AWS WWCO Professional Services, Amazon Web Services
- Raju Penmatcha - Senior Solutions Architect, AI Platforms, Amazon Web Services
- Jess Clark - Principal Security Engineer – Identity and Access Management, Amazon Web Services
- Emily Soward - Data Scientist, AWS WWCO Professional Services, Amazon Web Services
- Amit Lulla - Senior Solutions Architect ISV, AWS WWCS Geo Solutions Architect, Amazon Web Services
- Giuseppe Angelo Porcelli - Principal ML Solutions Architect, AI Platforms, Amazon Web Services
- Shelbee Eigenbrode - Principal AI/ML Specialist Solutions Architect, AWS WWSO AI/ML, Amazon Web Services
- Phi Nguyen – Front End Developer, AWS WWCO Professional Services, Amazon Web Services
- Jeremy Sobek - Technical Curriculum Developer, AWS T&C Curriculum, Amazon Web Services

Subject matter expert (SME) reviewers

- Mohammad Arbabshirani – Sr. Data Scientist Manager, AWS WWCO Professional Services, Amazon Web Services
- Chris Boomhower – Machine Learning Engineer, AWS WWCO Professional Services, Amazon Web Services
- Bruno Klein – Machine Learning Engineer, AWS WWCO Professional Services, Amazon Web Services
- Bruce Ross - Lens Leader, Senior Solutions Architect Well-Architected, Amazon Web Services

References

- AWS Architecture Center
- Architecture Best Practices for Machine Learning
- AWS Well-Architected Framework
- AWS Operational Excellence pillar
- AWS Security pillar
- AWS Reliability pillar
- AWS Performance Efficiency pillar
- AWS Cost Optimization pillar
- AWS Sustainability pillar
- Amazon AI Fairness and Explainability Whitepaper
- Overview of Deployment Options on AWS
- Crisp-DM 1.0
- Announcing New Tools for Building with Generative AI on AWS | Amazon Web Services
- Get started with generative AI on AWS using Amazon SageMaker JumpStart | Amazon Web Services

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Major update	Whitepaper updated to reflect current best practices , implementation guidance, and AWS services. Added the sustainability pillar that describes reducing your impact on the environme nt while taking advantage of AWS advanced AI/ML capabilities for your business. Updated technical reference s and resources including product documentation, blog posts, instructional and video links, and solution briefs.	July 5, 2023
Minor update	Fixed broken links.	April 6, 2023
Minor update	Corrected typo.	July 21, 2022
Major update	Whitepaper updated to reflect current best practices and AWS services.	October 12, 2021
Minor changes	Updated links in the whitepaper.	January 21, 2021
Initial publication	Machine Learning Lens first published.	April 16, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Best practices arranged by pillar

This is a list of best practices outlined in this paper organized by the pillars of the AWS Well-Architected Framework.

Operational excellence pillar

- MLOE-01: Develop the right skills with accountability and empowerment
- MLOE-02: Discuss and agree on the level of model explainability
- MLOE-03: Monitor model compliance to business requirements
- MLOE-04: Establish ML roles and responsibilities
- MLOE-05: Prepare an ML profile template
- MLOE-06: Establish model improvement strategies
- MLOE-07: Establish a lineage tracker system
- MLOE-08: Establish feedback loops across ML lifecycle phases
- MLOE-09: Review fairness and explainability
- MLOE-10: Profile data to improve quality
- MLOE-11: Create tracking and version control mechanisms
- MLOE-12: Automate operations through MLOps and CI/CD
- MLOE-13: Establish reliable packaging patterns to access approved public libraries
- MLOE-14: Establish deployment environment metrics
- MLOE-15: Enable model observability and tracking
- MLOE-16: Synchronize architecture and configuration, and check for skew across environments

Security pillar

- MLSEC-01: Validate ML data permissions, privacy, software, and license terms
- MLSEC-02: Design data encryption and obfuscation
- MLSEC-03: Ensure least privilege access
- MLSEC-04: Secure data and modeling environment
- MLSEC-05: Protect sensitive data privacy

- MLSEC-06: Enforce data lineage
- MLSEC-07: Keep only relevant data
- MLSEC-08: Secure governed ML environment
- MLSEC-09: Secure inter-node cluster communications
- MLSEC-10: Protect against data poisoning threats
- MLSEC-11: Protect against adversarial and malicious activities
- MLSEC-12: Restrict access to intended legitimate consumers
- MLSEC-13: Monitor human interactions with data for anomalous activity

Reliability pillar

- MLREL-01: Use APIs to abstract change from model consuming applications
- MLREL-02: Adopt a machine learning microservice strategy
- MLREL-03: Use a data catalog
- MLREL-04: Use a data pipeline
- MLREL-05: Automate managing data changes
- MLREL-06: Enable CI/CD/CT automation with traceability
- MLREL-07: Ensure feature consistency across training and inference
- MLREL-08: Ensure model validation with relevant data
- MLREL-09: Establish data bias detection and mitigation
- MLREL-10: Automate endpoint changes through a pipeline
- MLREL-11: Use an appropriate deployment and testing strategy
- MLREL-12: Allow automatic scaling of the model endpoint
- MLREL-13: Ensure a recoverable endpoint with a managed version control strategy

Performance efficiency pillar

- MLPER-01: Determine key performance indicators
- MLPER-02: Use purpose-built AI and ML services and resources
- MLPER-03: Define relevant evaluation metrics
- MLPER-04: Use a modern data architecture

- MLPER-05: Optimize training and inference instance types
- MLPER-06: Explore alternatives for performance improvement
- MLPER-07: Establish a model performance evaluation pipeline
- MLPER-08: Establish feature statistics
- MLPER-09: Perform a performance trade-off analysis
- MLPER-10: Detect performance issues when using transfer learning
- MLPER-11: Evaluate cloud versus edge options for machine learning deployment
- MLPER-12: Choose an optimal deployment option in the cloud
- MLPER-13: Evaluate model explainability
- MLPER-14: Evaluate data drift
- MLPER-15: Monitor, detect, and handle model performance degradation
- MLPER-16: Establish an automated re-training framework
- MLPER-17: Review for updated data/features for retraining
- MLPER-18: Include human-in-the-loop monitoring

Cost optimization pillar

- MLCOST-01: Define overall return on investment (ROI) and opportunity cost
- MLCOST-02: Use managed services to reduce total cost of ownership (TCO)
- MLCOST-03: Identify if machine learning is the right solution
- MLCOST-04: Tradeoff analysis on custom versus pre-trained models
- MLCOST-05: Use managed data labeling
- MLCOST-06: Use data wrangler tools for interactive analysis
- MLCOST-07: Use managed data processing capabilities
- MLCOST-08: Enable feature reusability
- MLCOST-09: Select optimal computing instance size
- MLCOST-10: Use managed build environments
- MLCOST-11: Select local training for small scale experiments
- MLCOST-12: Select an optimal ML framework
- MLCOST-13: Use automated machine learning
- MLCOST-14: Use managed training capabilities

- MLCOST-15: Use distributed training
- MLCOST-16: Stop resources when not in use
- MLCOST-17: Start training with small datasets
- MLCOST-18: Use warm-start and checkpointing hyperparameter tuning
- MLCOST-19: Use hyperparameter optimization technologies
- MLCOST-20 - Setup budget and use resource tagging to track costs
- MLCOST-21: Enable data and compute proximity
- MLCOST-22: Select optimal algorithms
- MLCOST-23: Enable debugging and logging
- MLCOST-24: Use appropriate deployment option
- MLCOST-25: Explore cost effective hardware options
- MLCOST-26: Right-size the model hosting instance fleet
- MLCOST-27: Monitor usage and cost by ML activity
- MLCOST-28: Monitor Return on Investment for ML models
- MLCOST-29: Monitor endpoint usage and right-size the instance fleet

Sustainability pillar

- MLSUS-01: Define the overall environmental impact or benefit
- MLSUS-02: Consider AI services and pre-trained models
- MLSUS-03: Select sustainable Regions
- MLSUS-04: Minimize idle resources
- MLSUS-05: Implement data lifecycle policies aligned with your sustainability goals
- MLSUS-06: Adopt sustainable storage options
- MLSUS-07: Define sustainable performance criteria
- MLSUS-08: Select energy-efficient algorithms
- MLSUS-09: Archive or delete unnecessary training artifacts
- MLSUS-10: Use efficient model tuning methods
- MLSUS-11: Align SLAs with sustainability goals
- MLSUS-12: Use efficient silicon
- MLSUS-13: Optimize models for inference

- MLSUS-14: Deploy multiple models behind a single endpoint
- MLSUS-15: Measure material efficiency
- MLSUS-16: Retrain only when necessary

Best practices by ML lifecycle phase

There are six phases in the machine learning lifecycle and within each there are six pillars of the Well-Architected Framework. The following index lists the best practices by lifecycle phase.

Business goal identification phase

Operational excellence pillar

- MLOE-01: Develop the right skills with accountability and empowerment
- MLOE-02: Discuss and agree on the level of model explainability
- MLOE-03: Monitor model compliance to business requirements

Security pillar

- MLSEC-01: Validate ML data permissions, privacy, software, and license terms

Reliability pillar

There are no reliability pillar best practices for business goal identification.

Performance efficiency pillar

- MLPER-01: Determine key performance indicators

Cost optimization pillar

- MLCOST-01: Define overall return on investment (ROI) and opportunity cost
- MLCOST-02: Use managed services to reduce total cost of ownership (TCO)

Sustainability pillar

- MLSUS-01: Define the overall environmental impact or benefit

ML problem framing phase

Operational excellence pillar

- MLOE-04: Establish ML roles and responsibilities
- MLOE-05: Prepare an ML profile template
- MLOE-06: Establish model improvement strategies
- MLOE-07: Establish a lineage tracker system
- MLOE-08: Establish feedback loops across ML lifecycle phases
- MLOE-09: Review fairness and explainability

Security pillar

- MLSEC-02: Design data encryption and obfuscation

Reliability pillar

- MLREL-01: Use APIs to abstract change from model consuming applications
- MLREL-02: Adopt a machine learning microservice strategy

Performance efficiency pillar

- MLPER-02: Use purpose-built AI and ML services and resources
- MLPER-03: Define relevant evaluation metrics

Cost optimization pillar

- MLCOST-03: Identify if machine learning is the right solution
- MLCOST-04: Tradeoff analysis on custom versus pre-trained models

Sustainability pillar

- MLSUS-02: Consider AI services and pre-trained models

- MLSUS-03: Select sustainable Regions

Data processing phase

Operational excellence pillar

- MLOE-10: Profile data to improve quality
- MLOE-11: Create tracking and version control mechanisms

Security pillar

- MLSEC-03: Ensure least privilege access
- MLSEC-04: Secure data and modeling environment
- MLSEC-05: Protect sensitive data privacy
- MLSEC-06: Enforce data lineage
- MLSEC-07: Keep only relevant data

Reliability pillar

- MLREL-03: Use a data catalog
- MLREL-04: Use a data pipeline
- MLREL-05: Automate managing data changes

Performance efficiency pillar

- MLPER-04: Use a modern data architecture

Cost optimization pillar

- MLCOST-05: Use managed data labeling
- MLCOST-06: Use data wrangler tools for interactive analysis
- MLCOST-07: Use managed data processing capabilities
- MLCOST-08: Enable feature reusability

Sustainability pillar

- MLSUS-04: Minimize idle resources
- MLSUS-05: Implement data lifecycle policies aligned with your sustainability goals
- MLSUS-06: Adopt sustainable storage options

Model development phase

Operational excellence pillar

- MLOE-12: Automate operations through MLOps and CI/CD
- MLOE-13: Establish reliable packaging patterns to access approved public libraries

Security pillar

- MLSEC-08: Secure governed ML environment
- MLSEC-09: Secure inter-node cluster communications
- MLSEC-10: Protect against data poisoning threats

Reliability pillar

- MLREL-06: Enable CI/CD/CT automation with traceability
- MLREL-07: Ensure feature consistency across training and inference
- MLREL-08: Ensure model validation with relevant data
- MLREL-09: Establish data bias detection and mitigation

Performance efficiency pillar

- MLPER-05: Optimize training and inference instance types
- MLPER-06: Explore alternatives for performance improvement
- MLPER-07: Establish a model performance evaluation pipeline
- MLPER-08: Establish feature statistics
- MLPER-09: Perform a performance trade-off analysis

- MLPER-10: Detect performance issues when using transfer learning

Cost optimization pillar

- MLCOST-09: Select optimal computing instance size
- MLCOST-10: Use managed build environments
- MLCOST-11: Select local training for small scale experiments
- MLCOST-12: Select an optimal ML framework
- MLCOST-13: Use automated machine learning
- MLCOST-14: Use managed training capabilities
- MLCOST-15: Use distributed training
- MLCOST-16: Stop resources when not in use
- MLCOST-17: Start training with small datasets
- MLCOST-18: Use warm-start and checkpointing hyperparameter tuning
- MLCOST-19: Use hyperparameter optimization technologies
- MLCOST-20 - Setup budget and use resource tagging to track costs
- MLCOST-21: Enable data and compute proximity
- MLCOST-22: Select optimal algorithms
- MLCOST-23: Enable debugging and logging

Sustainability pillar

- MLSUS-07: Define sustainable performance criteria
- MLSUS-08: Select energy-efficient algorithms
- MLSUS-09: Archive or delete unnecessary training artifacts
- MLSUS-10: Use efficient model tuning methods

Model deployment phase

Operational excellence pillar

- MLOE-14: Establish deployment environment metrics

Security pillar

- MLSEC-11: Protect against adversarial and malicious activities

Reliability pillar

- MLREL-10: Automate endpoint changes through a pipeline
- MLREL-11: Use an appropriate deployment and testing strategy

Performance efficiency pillar

- MLPER-11: Evaluate cloud versus edge options for machine learning deployment
- MLPER-12: Choose an optimal deployment option in the cloud

Cost optimization pillar

- MLCOST-24: Use appropriate deployment option
- MLCOST-25: Explore cost effective hardware options
- MLCOST-26: Right-size the model hosting instance fleet

Sustainability pillar

- MLSUS-11: Align SLAs with sustainability goals
- MLSUS-12: Use efficient silicon
- MLSUS-13: Optimize models for inference
- MLSUS-14: Deploy multiple models behind a single endpoint

Model monitoring phase

Operational excellence pillar

- MLOE-15: Enable model observability and tracking
- MLOE-16: Synchronize architecture and configuration, and check for skew across environments

Security pillar

- MLSEC-12: Restrict access to intended legitimate consumers
- MLSEC-13: Monitor human interactions with data for anomalous activity

Reliability pillar

- MLREL-12: Allow automatic scaling of the model endpoint
- MLREL-13: Ensure a recoverable endpoint with a managed version control strategy

Performance efficiency pillar

- MLPER-13: Evaluate model explainability
- MLPER-14: Evaluate data drift
- MLPER-15: Monitor, detect, and handle model performance degradation
- MLPER-16: Establish an automated re-training framework
- MLPER-17: Review for updated data/features for retraining
- MLPER-18: Include human-in-the-loop monitoring

Cost optimization pillar

- MLCOST-27: Monitor usage and cost by ML activity
- MLCOST-28: Monitor Return on Investment for ML models
- MLCOST-29: Monitor endpoint usage and right-size the instance fleet

Sustainability pillar

- MLSUS-15: Measure material efficiency
- MLSUS-16: Retrain only when necessary

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.